
Departamento de Tecnología Electrónica

PROYECTO FIN DE CARRERA

# DISEÑO E IMPLEMENTACIÓN EN VHDL DEL JUEGO 'HUNDIR LA FLOTA' CON COMUNICACIÓN POR INFRARROJOS.

Autor: Jorge Balsa González

Tutora: Marta Portela García

Leganés, octubre de 2012

Título: Diseño e implementación en VHDL del juego ‘Hundir la flota’ con comunicación por infrarrojos

Autor: Jorge Balsa González

Director:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 31 de octubre de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

A mi familia y mi novia.

Muchas gracias, os quiero.

# AGRADECIMIENTOS

Me gustaría agradecer todo el apoyo recibido por parte de mi tutora Marta, su paciencia y confianza en mí han hecho posible este proyecto.

A mis padres, las personas que más quiero y aprecio, por los valores que me han inculcado. A mi madre porque la considero la mujer más especial y trabajadora del mundo, se lo debo todo. A mi padre porque a pesar de los palos que nos pueda dar la vida, siempre me ha enseñado a luchar.

A mi hermano, el mayor referente en mi vida, por mostrarme siempre el camino. Y a mi hermana, cuyos valores son envidiables, por ser tan especial y estar siempre ahí.

A mi novia, a la que el destino me ha devuelto, por su confianza ciega en mí, por su apoyo y amor infinito, y por ser simplemente…todo lo que necesito.

A mis amigos de la universidad, Chava, Mario, Julián, Heras, Tocho, Cuevas y muchos más, por recorrer esta aventura juntos y por todos los momentos inolvidables que hemos vivido y viviremos.

Y por último a mis abuelos, que allá donde estén les hubiera encantado saber que su nieto lo ha conseguido.

# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# CAPÍTULO 1

## 1 INTRODUCCIÓN

El diseño digital tiene una gran relevancia en el mundo actual. En los últimos años ha aumentado considerablemente este tipo de diseños, debido al amplio abanico de funcionalidades que se le pueden dar a los circuitos digitales. Tal es la importancia, que forma parte del contenido en asignaturas impartidas en diversos estudios universitarios. Como es el caso de la asignatura "Laboratorio de Microelectrónica", que se imparte en 4º curso de Ing. de Telecomunicación por el departamento de Tecnología Electrónica. Es una asignatura eminentemente práctica. En ella, se pretende dar una perspectiva práctica del diseño de un circuito digital integrado de aplicación específica, utilizando lenguajes de descripción hardware de alto nivel. Cada año se realizan unas prácticas en las cuáles los alumnos deben de desarrollar proyectos utilizando el lenguaje de descripción de hardware VHDL, y cuya implementación final se realiza sobre una FPGA. Tradicionalmente estos proyectos han consistido en juegos como las 4 en Raya, el juego de La Oca o Las Damas.

En este proyecto final de carrera se pretende estudiar y desarrollar un sistema digital de complejidad media/alta programado en una FPGA, cuya función principal sea permitir jugar al conocido juego Hundir la Flota. Para ello, se aplicarán los conocimientos adquiridos en la asignatura "Tecnología Electrónica II" que se imparte en el 3º curso de la titulación Ingeniería Industrial por el departamento de Tecnología Electrónica. Y se utilizará VHDL para la totalidad del código del sistema. Se unificará el diseño en VHDL con conocimientos para diseñar circuitos electrónicos y de este modo, obtener los objetivos de este proyecto.

El juego debe desarrollarse para poder aplicarlo en la asignatura "Laboratorio de Microelectrónica", como contenido para las prácticas de esta asignatura. Con el fin de ampliar el repertorio de juegos y poder alternarlos de un año a otro.

### *1.1 Objetivos*

Los objetivos del proyecto se describen a continuación:

- Diseñar e implementar un circuito en una placa FPGA con el lenguaje de descripción de hardware VHDL, que permita jugar al juego Hundir la Flota. El juego tiene que cumplir con las reglas del mismo y simular al original.

- El juego tiene que ser interactivo. Debe estar diseñado para que un jugador, a través de la interfaz con su propio ordenador y en función de sus movimientos, pueda comunicarse con otro jugador, que a su vez dispone de otra interfaz y otro ordenador, y puedan disputar una partida hasta que se proclame un campeón. Será necesario especificar los componentes hardware a utilizar para realizar la implementación de dicha funcionalidad.

- Estudio de viabilidad para poder aplicar este proyecto como contenido de prácticas en "Laboratorio de Microelectrónica", en función de las horas disponibles para ello (30h) y del tamaño de los grupos de trabajo (2 personas por grupo).

- Realizar una propuesta sobre el contenido que debería considerarse dentro de las prácticas obligatorias para el aprobado, y qué contenido debería considerarse para subir nota.

## *1.2 Estructura de la memoria*

Para facilitar la lectura de la memoria y la búsqueda de información concreta dentro de ella, se detalla a continuación un breve resumen del contenido de cada capítulo.

Capítulo 2: En este capítulo se detallan las especificaciones que se requieren cumplir para este proyecto. Se explican las reglas y el funcionamiento del juego que se quiere diseñar. Además del hardware que va a ser necesario para conseguirlo. Por último, se explica el protocolo que se utiliza para la comunicación infrarroja.

Capítulo 3: En este capítulo se muestra una visión completa del circuito diseñado. El sistema global está dividido en bloques y se explica detalladamente cómo funcionan, y qué función desempeñan cada uno de ellos. Se explican todas las máquinas de estados y las entidades que forman el sistema.

Capítulo 4: En este capítulo se demuestra el correcto funcionamiento del juego. Se realizan una serie de comprobaciones que simulan las posibles opciones a realizar en el sistema. La validación está acompañada de fotos reales del juego donde se aprecia que su simulación es correcta y válida.

Capítulo 5: En este capítulo se explican las conclusiones obtenidas después de realizar el proyecto. Se explica si se han conseguido los objetivos y se exponen los trabajos futuros a realizar en otros proyectos.

Capítulo 6: En este capítulo se presenta el presupuesto del proyecto. Se detalla un presupuesto con el coste del personal, el coste del material, el coste del software utilizado y los costes indirectos.

# CAPÍTULO 2

# 2 ESPECIFICACIONES DEL DISEÑO

## *2.1 El juego Hundir la Flota*

Hundir la Flota es un juego para 2 jugadores, que se basa en intentar hundir todos los barcos del enemigo, acertando las casillas en las que el oponente ha colocado sus barcos. El tablero del juego se divide a su vez en 2 pequeños tableros cuadriculados que representan el mapa de cada jugador. Cada tablero tiene 8 filas y 8 columnas. Cada posición se denomina con sus coordenadas (Fila, Columna). Para referirse en esta memoria a filas y a columnas se hará con la nomenclatura que se indica en la Figura 1.

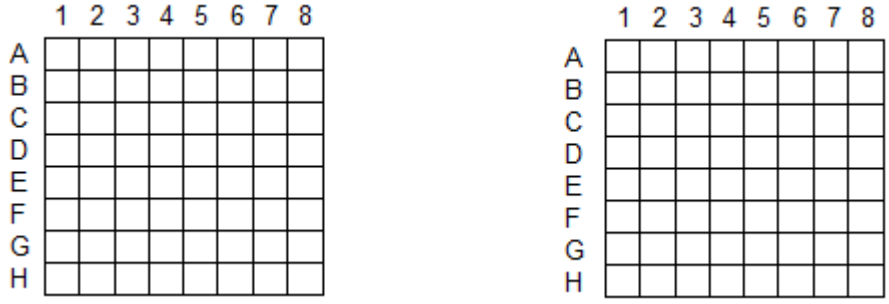



Figura 1. Tablero del juego.

El tablero de la izquierda corresponde al mapa del propio jugador, en el cuál el jugador coloca sus barcos y registra las bombas del oponente. En el otro, se registra el lanzamiento de las bombas propias. Antes de comenzar, cada jugador posiciona los barcos de forma secreta o invisible al oponente. Los barcos se pueden colocar horizontal o verticalmente. De esta forma, no se permiten lugares solapados, ya que cada uno ocupa posiciones únicas. Ambos participantes poseen y deben ubicar igual número de barcos. Cada jugador dispone de la siguiente flota:

- 1 barco portaaviones (5 casillas)
- 1 barco acorazado (4 casillas)
- 2 barcos crucero (3 casillas)
- 1 barco destructor (2 casillas)

Una vez todos ellos han sido posicionados, se inicia una serie de rondas. En cada ronda, cada jugador indica en su turno una posición del tablero de su oponente. Si esa posición es ocupada por una parte de un barco, el oponente indica *tocado.* Cuando todas las posiciones de un mismo barco han sido *tocadas* se le indica *hundido,* remarcando la importancia que conlleva una nave destruida. Ahora bien, si la posición indicada no posee un barco alojado, se indica con *agua*. La forma de pasar el turno es circular, pasa de un jugador a otro con cada turno. Cada turno consiste en el lanzamiento de una bomba. El ganador es el primer jugador que consigue hundir todos los barcos de su oponente. El juego termina cuando un jugador gana.

Una vez conocidas las reglas, se ha realizado alguna modificación para adaptarlo al proyecto. A continuación se describen las especificaciones sobre la funcionalidad que se han establecido para el sistema a desarrollar en este proyecto. Cabe destacar la flota de la que dispone cada jugador:

- 1 barco grande (4 casillas)
- 3 barcos medianos (3 casillas)
- 2 barcos pequeños (2 casillas)

La Figura 2 muestra el tablero al comienzo de una partida. Debajo del tablero de la izquierda aparecen los barcos que se deben colocar. La selección se realiza moviendo el cursor que viene indicado por una flecha a la izquierda de los barcos. A su vez se indica con un contador el número de barcos que faltan por situar en el tablero. La forma de reflejar el cambio de turno se indica en el tablero mediante un cuadrado que cambia de color (verde=tu turno, rojo=turno del oponente).

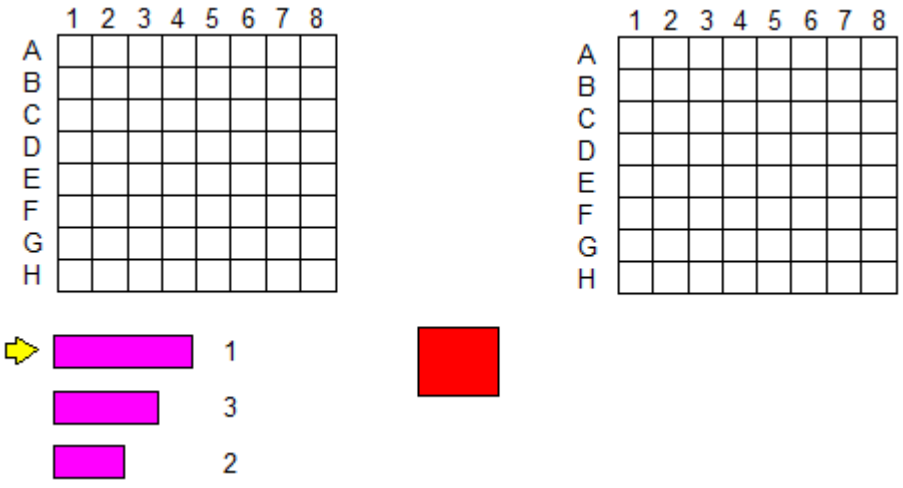



Figura 2. Tablero del juego al comenzar.

El turno de inicio se determina a partir de la colocación de los barcos. El jugador que primero coloque sus barcos, es el primero en comenzar la partida. Por otro lado, cuando un jugador indica que ha sido *tocado*, al atacante se le marca con rojo esa posición. Un barco *hundido* se indica con negro. Por último, un lanzamiento sin acierto es reflejado en el tablero en azul, indicando *agua.* La partida termina cuando un jugador hunde todos los barcos del oponente y finalmente, aparece un mensaje en la pantalla indicando si has sido el vencedor o el perdedor de la partida.

## 2.2 Hardware a utilizar

Una vez conocidas las reglas del juego, es necesario conocer el hardware disponible en el laboratorio que puedan ser de utilidad y determinar qué componentes se usarán en la implementación finalmente.

En primer lugar, teniendo en cuenta que se va a diseñar un juego interactivo, se va a necesitar un monitor de ordenador, una interfaz para el jugador y un cable de conexión entre ambas. En el laboratorio se dispone de *monitores con interfaz VGA* que se utilizarán para visualizar la partida puesto que la placa FPGA que se va a utilizar, y que se describe en detalle en el apartado 2.4, incluye un conector VGA.

Para la interfaz del jugador se dispone de un teclado matricial que podría conectarse a la placa FPGA, y de interruptores y pulsadores de la placa FPGA suministrada para este proyecto. Por simplicidad y para evitar tener más dispositivos, se ha decidido utilizar los componentes de la placa FPGA, por lo que será necesario diseñar el hardware capaz de transformar la activación/desactivación de los interruptores en movimientos para el juego.

En segundo lugar, el juego debe tener un mecanismo de comunicación para que los 2 jugadores puedan interactuar. Como se parte de que el autor del proyecto ya ha diseñado un *receptor de infrarrojos* para un trabajo dirigido, se ha decidido diseñar una comunicación por infrarrojos. Por lo tanto, se va a necesitar el circuito *receptor*, que cumple con el protocolo RC5 de Philips, y diseñar un *emisor de IR* que también lo cumpla. Para ello es necesaria una *placa protoboard*, disponible en el laboratorio, como base del montaje hardware de los circuitos infrarrojos. Se van a necesitar cables, resistencias, transistores y todo tipo de material electrónico que se incluya en el diseño. Como *receptor de IR* se va a utilizar un dispositivo SFH 5110 y para el *emisor* hay que realizar un estudio de las características necesarias para seleccionar el más adecuado. Hasta el momento en el que esté diseñado, se va a utilizar un *mando a distancia universal* simulando las señales que envía el *emisor* lo que ayudará en tareas de depuración. Por último, en un *osciloscopio* de los que se

dispone en el laboratorio se pueden observar las ondas infrarrojas y estudiar su comportamiento.

### 2.2.1 VGA

Un monitor VGA se controla por medio de tres señales analógicas, que especifican la intensidad de los colores (RGB, Red Green Blue), y dos señales de sincronismo, horizontal (HSYNC) y vertical (VSYNC). Ver referencia [1].

Para dibujar una imagen en la pantalla, hay que realizar un barrido de líneas, empezando por la esquina superior izquierda y barriendo hacia la derecha y hacia abajo. Es decir, el monitor va dibujando los datos en líneas horizontales. Después de cada línea horizontal es necesario dar un pulso a la señal de sincronismo horizontal HSYNC, y después de cada pantalla, cuando están dibujadas todas las líneas horizontales, hay que dar un pulso a la señal de sincronismo vertical VSYNC. Para implementar el barrido de cada línea, se utiliza un contador, que se reinicia al comienzo de cada línea, y que permite temporizar la señal HSYNC apropiadamente. En la Figura 3 se muestran las formas de onda de un barrido de línea horizontal.

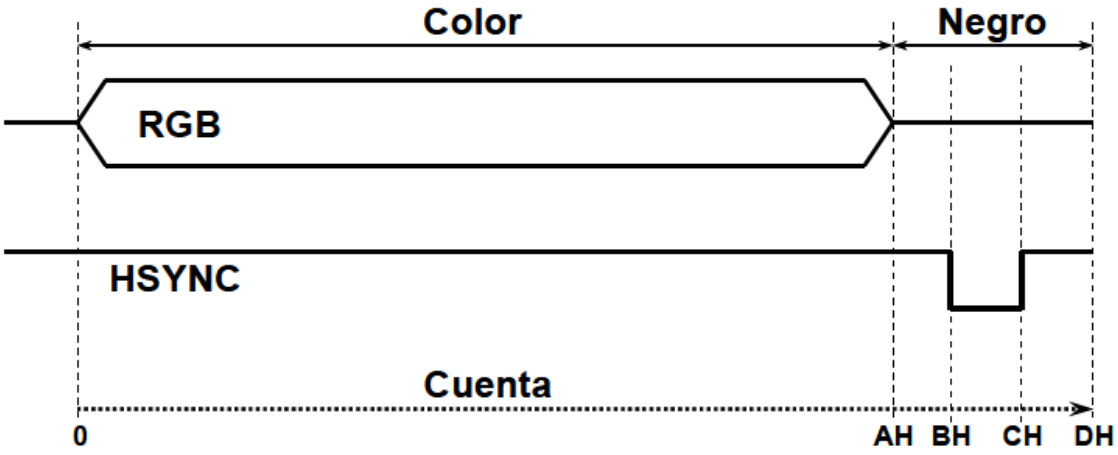



Figura 3. Formas de onda de un barrido de línea horizontal[1].

El proceso de barrido de línea se repite tantas veces como líneas haya en la pantalla, y después se procede a realizar el sincronismo vertical. La señal VSYNC tiene una forma equivalente a HSYNC, pero el pulso se produce tras haberse barrido todas las líneas horizontales, como puede observarse en la Figura 4. Para implementar el barrido de líneas, se utiliza otro contador, que se reinicia al comienzo de cada pantalla, y que permite temporizar la señal VSYNC.

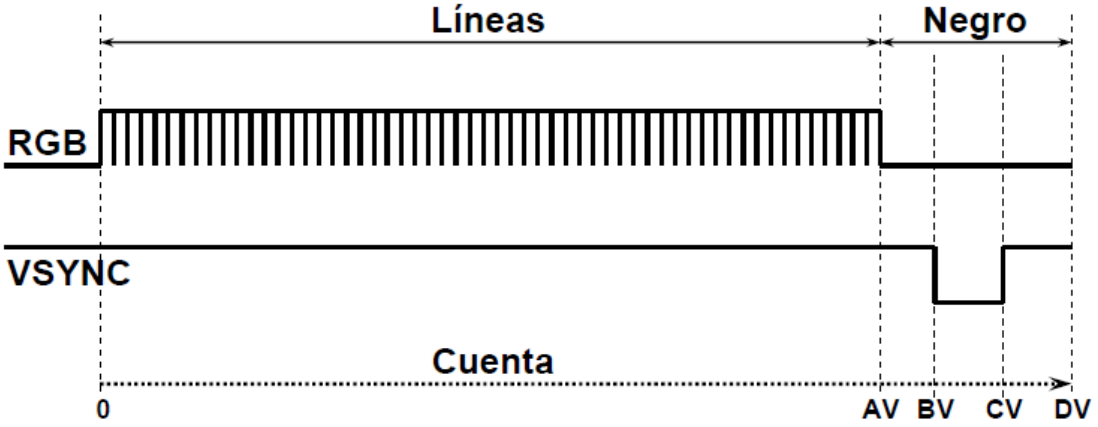



Figura 4. Formas de onda de un barrido de pantalla[1].

La resolución que se va a utilizar es de 640x480, con una frecuencia de actualización de 60Hz. Para esta resolución y suponiendo una frecuencia de reloj de 25MHz (la mitad del reloj de la *placa FPGA*), los tiempos correspondientes a cada evento y los valores correspondientes de los contadores se muestran en la Tabla 1.

| Horizontal | | | Vertical | | |
|---|---|---|---|---|---|
| | Tiempo | Puntos a recorrer | | Tiempo | Líneas a recorrer |
| **AH** | 25,60 us | 640 | **AV** | 15,360 ms | 480 |
| **BH** | 26,16 us | 654 | **BV** | 15,680 ms | 490 |
| **CH** | 30,08 us | 752 | **CV** | 15,744 ms | 492 |
| **DH** | 32,00 us | 800 | **DV** | 16,672 ms | 521 |

Tabla 1. Tiempos de sincronismo.

Se observa que los valores de cuenta en los puntos AH y AV son 640 y 480 respectivamente, y se corresponden con el número de píxeles de anchura y altura de la pantalla. Por tanto los valores de los contadores representan las coordenadas X e Y del píxel que se está representando en un momento dado. Se observa también que DV es el periodo de la frecuencia de actualización (1/60).

Puesto que la *placa FPGA* tiene un reloj de 50 MHz y los cálculos de tiempo están realizados para 25 MHz, los contadores sólo deberán activarse uno de cada dos ciclos de reloj.

### 2.2.2 Placa FPGA

La *placa FPGA* es el componente con más trascendencia en el diseño. Se ha proporcionado para el proyecto el dispositivo "xc3s500e-4ft256" de la familia Spartan-3. En la Figura 5 se muestra este componente. Ver referencia [4].

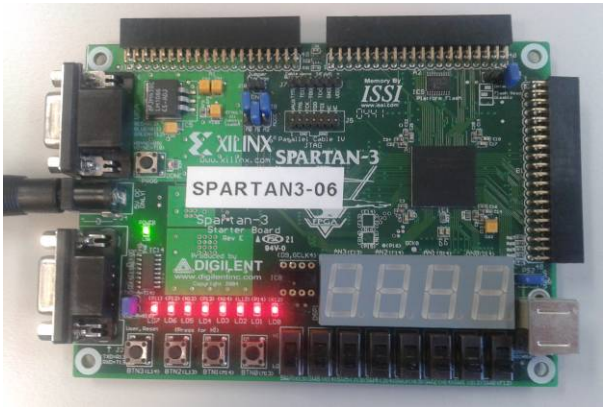



Figura 5. Placa FPGA.

A continuación se detallan sus características y la funcionalidad que ofrece:

- La *placa* tiene conexión *VGA*. Hay que diseñar un bloque que controle la interfaz *VGA* explicada anteriormente para que al conectarla con el cable al *monitor* se pueda visualizar el contenido del juego.

- Posee 6 *interruptores* y 4 *pulsadores.* Hay que diseñar la interfaz para conseguir interpretar el pulsado de estos botones y transformarlos en movimientos, dentro de la partida, de los cursores, del giro en vertical/horizontal de los barcos y una señal de confirmación 'intro' para seleccionar, colocar, disparar, etc. Van a funcionar como interfaz para que el jugador pueda interactuar en el juego.

- Dentro de la *placa* está la *FPGA,* es el dispositivo donde se implementará el diseño a desarrollar.

- A su vez la placa dispone de unos diodos *LED.* Se van a utilizar para poder interpretar visualmente el significado de determinadas señales enviadas por infrarrojos a lo largo del juego. Y para saber si se han enviado correctamente o se ha producido algún tipo de error.

- El código de descripción de hardware se diseña con una herramienta software de desarrollo compatible con las FPGAs de Xilinx, ISE (Integrated Software Environment). Ver referencia [2]. Y para realizar simulaciones se utiliza la herramienta ModelSim. Ver referencia [3].

## 2.3 Protocolo RC5 de Philips

El protocolo RC5 de Philips [10] utiliza la codificación tipo Manchester o codificación de doble fase. Dentro de cada bit se produce una transición en el medio, por lo que se alterna de nivel bajo a nivel alto, en el caso de que el bit tenga un valor lógico 1, o de nivel alto a nivel bajo, en el caso de que el valor lógico del bit sea 0. La frecuencia de la onda portadora es de 36KHz. Por lo tanto, el protocolo va a consistir en un tren de pulsos cuadrados de 36KHz, cuyo periodo es la inversa de esta frecuencia, 27,7us. Sabiendo que la mitad de un bit está formado por 32 pulsos, y por lo tanto, que un bit completo lo forman 64 pulsos, se le asigna una duración siempre constante de 1,778ms. Donde un 0 lógico está codificado como 889us de duración de los 32 pulsos y 889us de reposo. Y al contrario con un 1 lógico. En la Figura 6 se observan los dos posibles valores de un bit y su representación.


Figura 6. Bit protocolo RC5 Philips.

El código de transmisión es una palabra que consta de 14 bits. El primer bit enviado se llama bit *Start* o bit de Inicio, y tiene como función indicar que comienza una nueva trama. Su valor lógico siempre será 1. El segundo bit, el bit *Field* o bit de campo, selecciona entre los comandos 0 y 63 de un total de 128 si su valor lógico es 0, o entre los comandos 64 y 127 si su valor lógico es 1. El tercer bit enviado en cada transmisión es el bit *C*. Este bit es muy importante ya que tiene como misión indicar que se ha pulsado de nuevo un botón, y sirve para distinguir las transmisiones enviadas con una sola pulsación, de las transmisiones múltiples. El bit C alterna su valor lógico con cada nueva pulsación. A continuación se envían 5 bits, que forman una de las posibles 32 direcciones que existen en el protocolo RC5. Las direcciones denotan si se quiere activar la TV, el video, etc. Por último, los siguientes 6 bits componen la palabra completa que se envía en cada transmisión. Éstos son los bits de comando y determinan los 128 posibles comandos que existen en este protocolo.

La longitud completa de la trama es igual a 14 * 1,778ms= 24.892 ms. Si la tecla se mantiene presionada, la trama se reenvía continuamente, pero con una pausa equivalente a 50 bits (50 * 1.778ms = 88.900ms) entre el final de una transmisión y

el inicio de otra. Por lo tanto, cada vez que se envía una transmisión con este protocolo, se envía una palabra de 14 bits como la que se observa en la Figura 7.

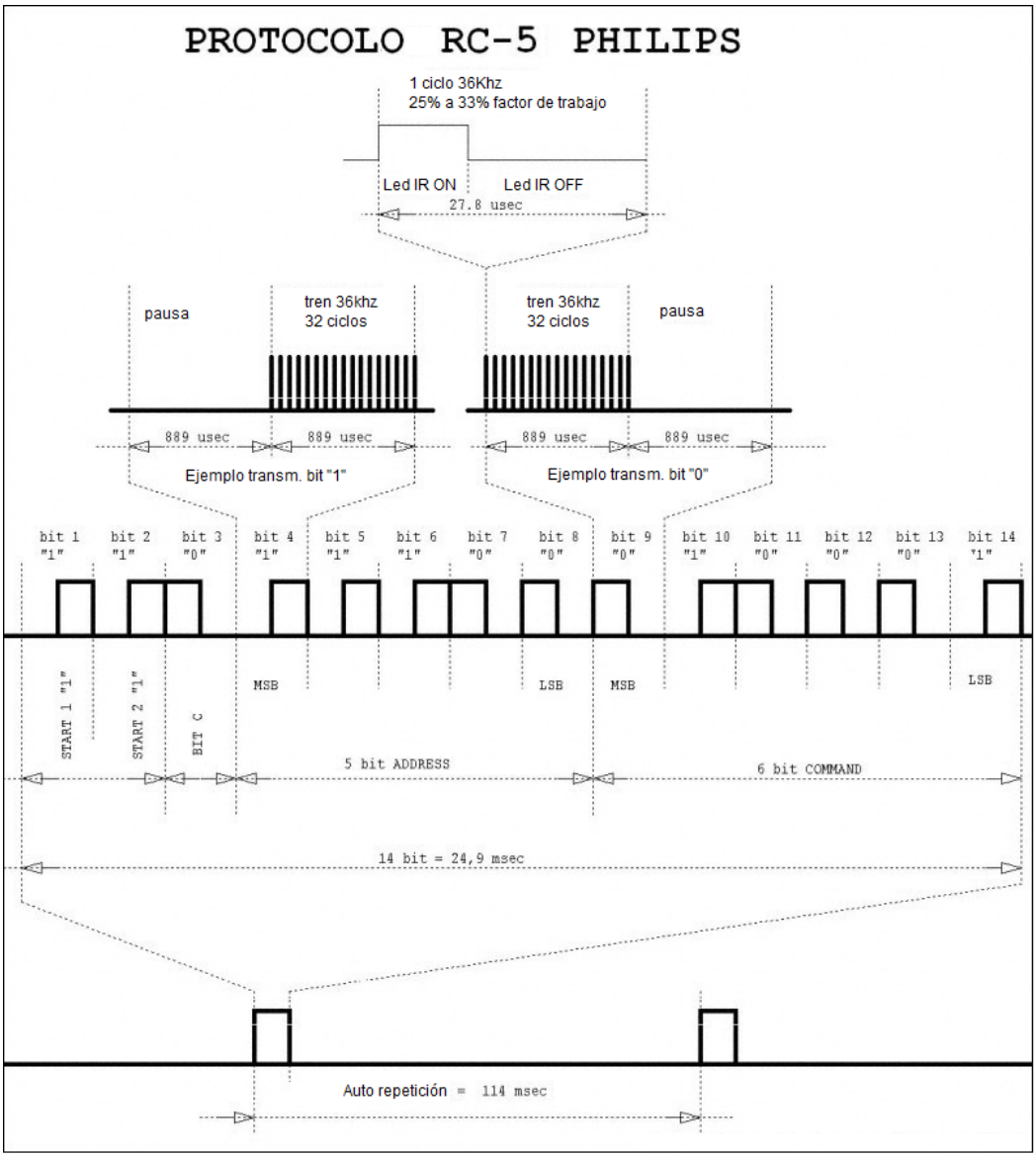



Figura 7. Protocolo RC5 Philips.

Hay que seleccionar un dispositivo emisor de infrarrojos que cumpla con este protocolo y se comunique con el receptor de infrarrojos ya diseñado. Además, hay que diseñar los circuitos de acondicionamiento de los sensores infrarrojos para el correcto funcionamiento de éstos.

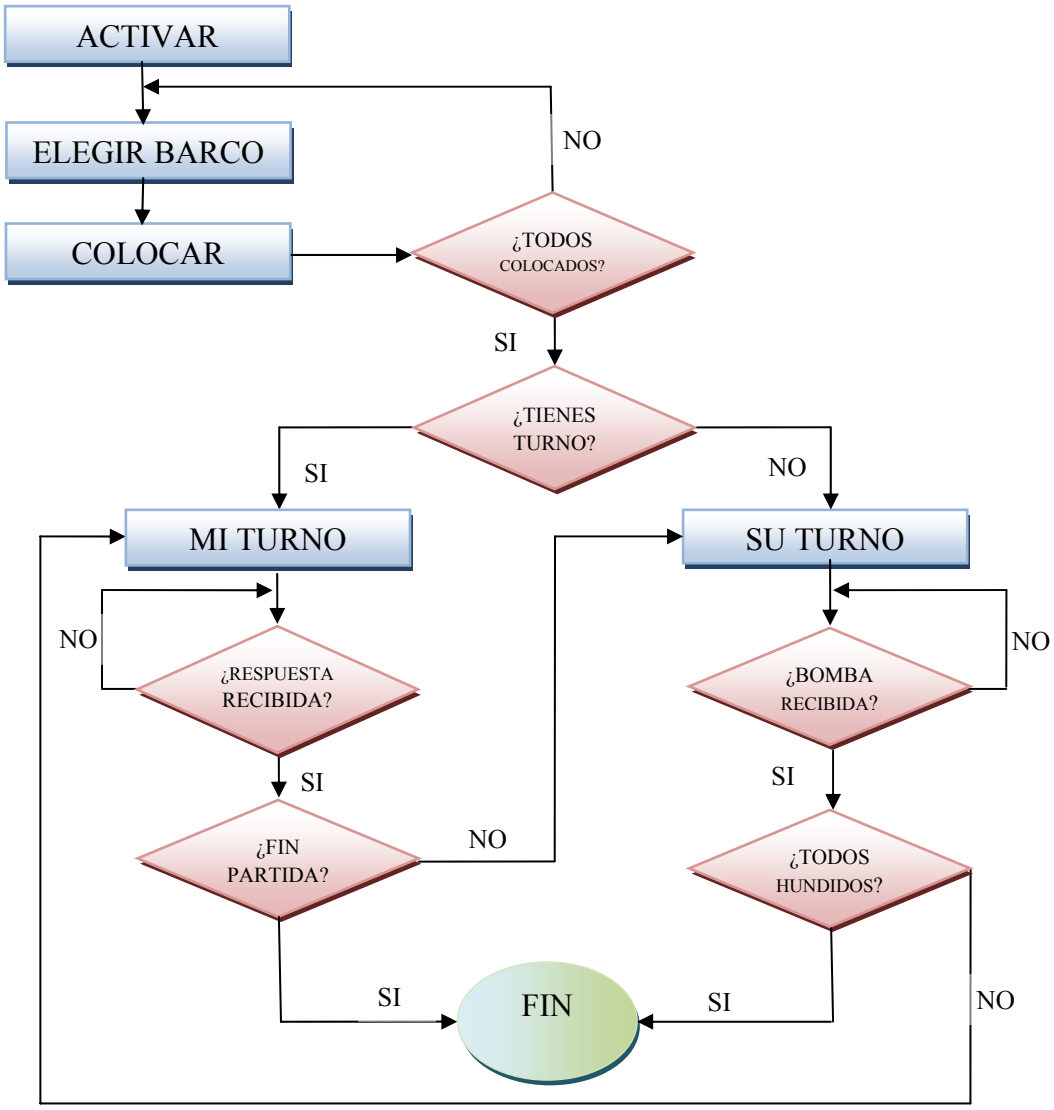
Por último, este juego está diseñado para que jueguen 2 jugadores, por lo que a la hora de la práctica real, se necesitarían dos unidades de cada componente hardware expuesto anteriormente.

# CAPÍTULO 3

## 3 ESTRUCTURA DEL DISEÑO

Hay que diseñar un sistema que cumpla con las funciones especificadas en el capítulo anterior. El sistema se divide en una serie de bloques. En primer lugar, en este capítulo se muestra una visión global del circuito completo para comprender el sistema diseñado. Una vez se tenga una idea del circuito, se va a entrar a detallar cada bloque por separado para entender su funcionamiento.

A continuación se muestra un diagrama de flujo con los pasos a seguir en el juego según el diseño realizado.



Es necesario activar el juego mediante un interruptor para comenzar la partida. En la Figura 8 se muestra la apariencia del juego al empezar.

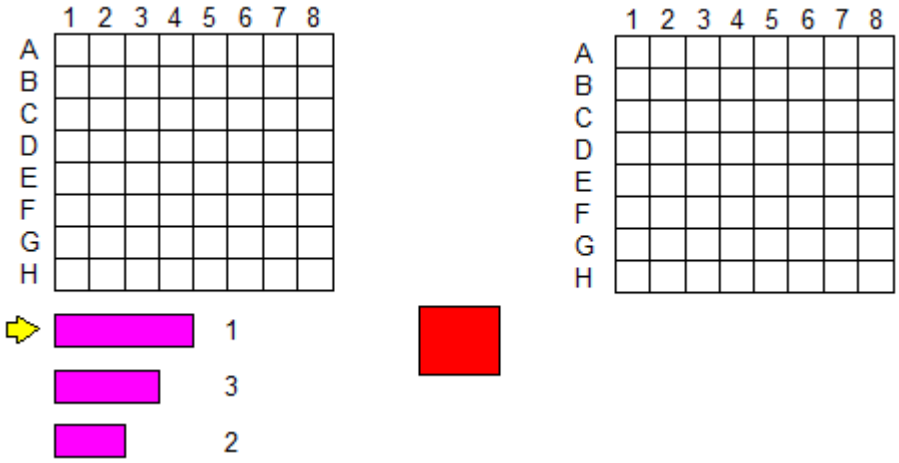



Figura 8. Estado inicial del juego.

Una vez se activa el juego, aparece el puntero (flecha) para elegir el barco que se quiera colocar. Se selecciona el barco y aparece un nuevo puntero en el tablero de la izquierda, correspondiente al mapa del propio jugador. Es posible desplazarse por todo el tablero y colocar el barco en la posición que se quiera. El sistema analiza la posición seleccionada y comprueba si se puede colocar la nave. Se analiza la posición del barco al colocarlo, si está horizontal o vertical, y si se saldría de los límites del tablero debido al número de casillas del barco. Otro dato a analizar, comprobando la memoria RAM, es si en alguna de las casillas, donde iría el barco, ya estuviese ocupada por otra nave. Una vez colocado el barco, se repite todo el proceso hasta terminar de colocar toda la flota. En la Figura 9 se muestra el estado de la partida cuando todos los barcos han sido colocados.

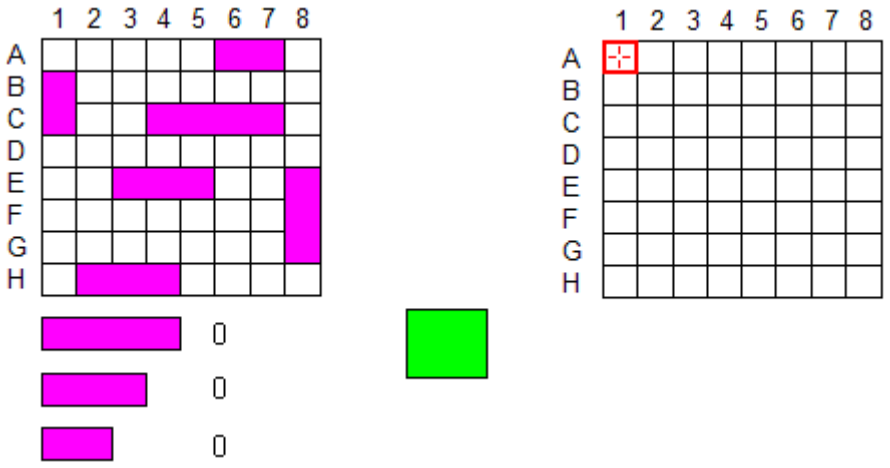



Figura 9. Flota colocada y lista para jugar.

El primer jugador en acabar de colocar su flota, envía una señal IR al oponente. Esta señal va a determinar el primer turno de la partida. El sistema del jugador que recibe la señal se ve alterado, y una vez ha terminado de colocar su flota se mantiene en una espera, respetando el turno. El jugador que ha terminado primero, comienza a lanzar la primera bomba de la partida, ya que posee el turno inicial. En esta parte del juego, el jugador que tiene el turno se mueve con otro puntero por el tablero de la derecha, correspondiente al mapa de su oponente.

Una vez ha elegido la casilla donde desea disparar la bomba, se comprueba que no haya una bomba anterior colocada en esa casilla. Si esto es así, se debe elegir otra casilla. En el caso de que sea correcta la selección, se envían por infrarrojos las coordenadas de la posición donde se ha depositado la bomba. El sistema del jugador oponente recibe e interpreta esas señales, y envía una respuesta con el resultado del lanzamiento (agua, tocado o hundido). El turno cambia al otro jugador, y realiza el mismo procedimiento anterior: lanzar bomba y esperar respuesta. En la Figura 10 se muestra el estado de la partida tras una serie de lanzamientos por parte de ambos jugadores.

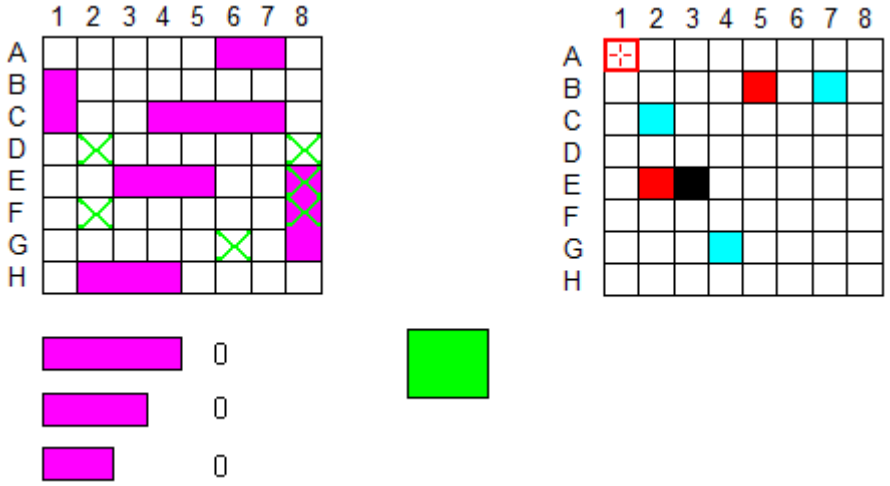



Figura 10. Tablero tras varios lanzamientos de bombas.

Los turnos se alternan continuamente hasta que uno de los jugadores sea capaz de hundir toda la flota enemiga. Cuando esto ocurre, el sistema del jugador que detecta que todos sus barcos han sido eliminados envía una señal IR indicando el final de la partida. Finalmente, a cada jugador le aparece un mensaje en la pantalla en función de lo que haya hecho, ganar o perder, junto con un emoticono alegre o triste, respectivamente.

## 3.1 Diagrama de bloques del circuito completo

El sistema ha sido diseñado bloque a bloque teniendo especial cuidado en que se cumpliesen las especificaciones asociadas a cada uno de ellos. En la Figura 11 se muestra el diagrama de bloques del circuito completo.

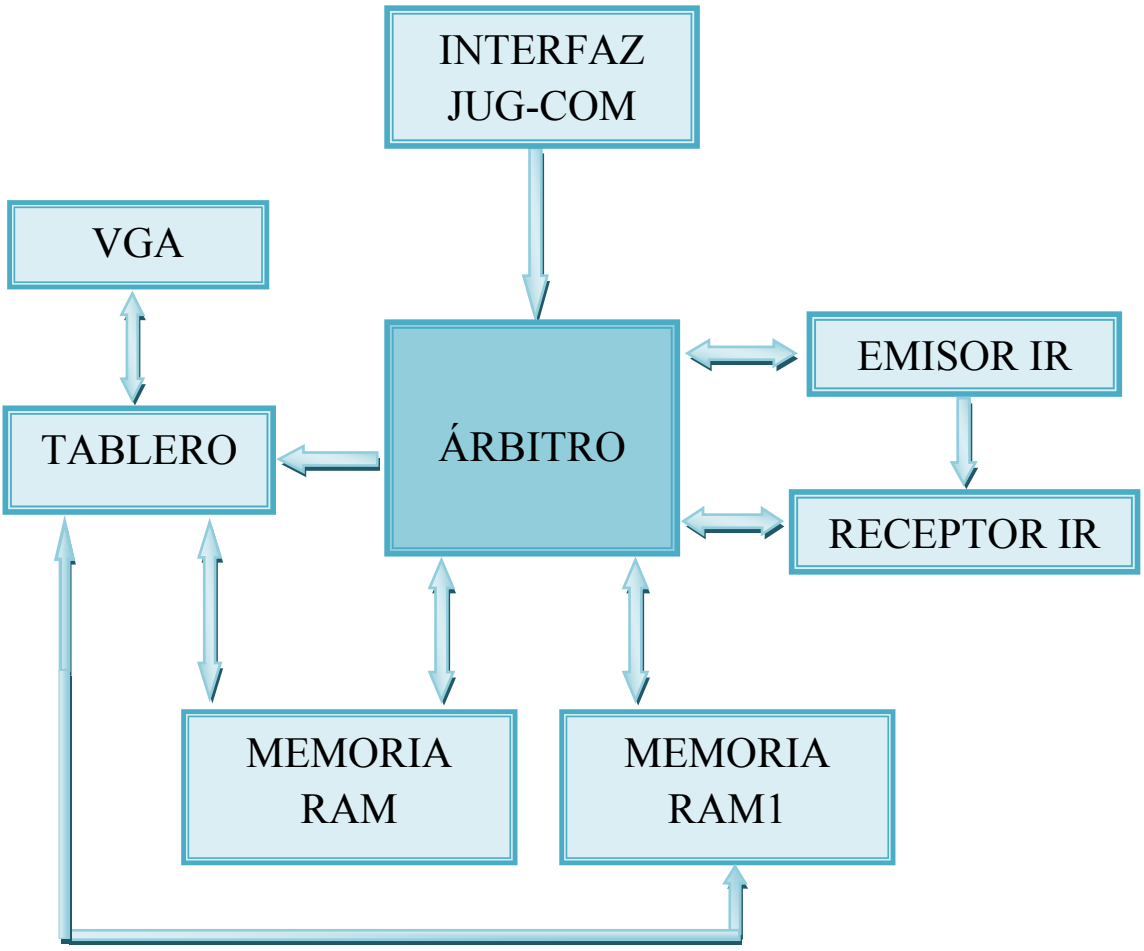



Figura 11. Diagrama de bloques del circuito completo.

Estos son los bloques integrados en el circuito. El sistema está formado por un bloque principal llamado *Árbitro* y 7 bloques más, correspondientes a la comunicación infrarroja, la memoria, el tablero y la salida por pantalla del juego. Cada uno de ellos se desarrolla según las funciones que tiene que cumplir. A continuación se explica detalladamente el funcionamiento y la entidad de cada uno de ellos, y lo que aportan al sistema.

## 3.2 Funcionalidad e interfaz de cada componente

### 3.2.1 Árbitro

El *Árbitro* es el bloque de control del sistema, y por lo tanto, está considerado como el más importante. Se comunica con el resto de bloques del circuito exceptuando la *VGA*. Es el encargado de ejecutar una gran variedad de acciones, todas ellas asociadas al resto de bloques. Su diseño está formado por una máquina de estados (FSM) principal y tres máquinas secundarias (asociadas a alguno de los estados de la principal), además de un gran número de procesos complejos.

La FSM principal controla el juego en general, frente a las secundarias que controlan determinadas partes del juego. A continuación se detallan las FSMs y se explica el funcionamiento de cada una de ellas por separado. En la Figura 12 se muestra la FSM principal.

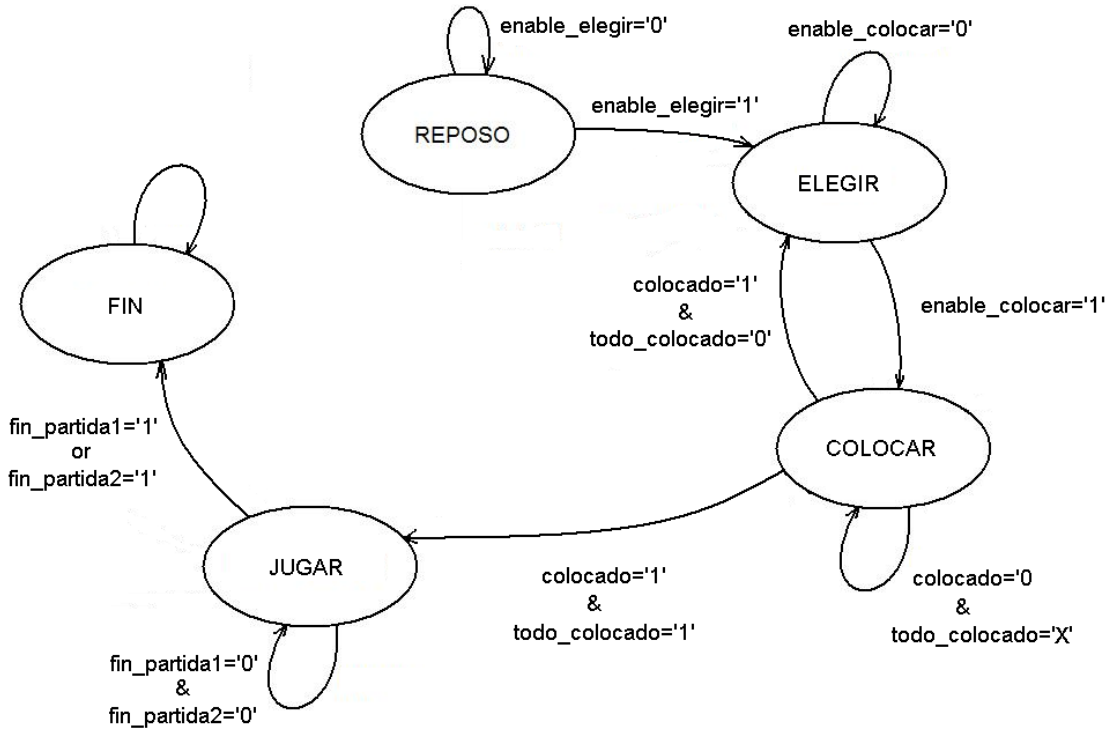



Figura 12. Máquina de estados principal del árbitro.

La FSM principal parte de un estado de *Reposo,* en el cuál todos los valores están inicializados, y el sistema está esperando a que se habilite una señal para comenzar la partida. El jugador habilita esta señal mediante la activación de un interruptor. El juego comienza a funcionar y todas las señales toman su valor por defecto. El sistema pasa al estado *Elegir,* donde el jugador tendrá la opción de elegir

el barco que quiera colocar. En este estado, el sistema principal se ramifica a una FSM secundaria, correspondiente a la elección de la flota. Esta máquina se va a llamar FSM secundaria Nº1, mostrada en la Figura 13 y se explicará más adelante. Cuando se ha seleccionado un barco, la máquina principal cambia al estado *Colocar*, en el cuál el jugador debe colocar el barco seleccionado en su tablero. En este estado, el sistema principal vuelve a ramificarse a otra FSM secundaria, nombrada como FSM secundaria Nº2, que se muestra en la Figura 14 y que también se detallará más adelante en este capítulo. El sistema alterna de un estado a otro hasta que se seleccionan y colocan todos los barcos que forman la flota. Con todos los barcos colocados por parte de los dos jugadores, el juego está listo para comenzar a disputar la partida. Por ello, la máquina pasa al estado *Jugar.* Es el momento de moverse por todo el tablero del oponente buscando la casilla que resulte tener un barco depositado en ella, lanzar bombas e intentar hundir a toda la flota enemiga. Como ocurre en otros estados de la máquina principal, se ramifica en una tercera FSM secundaria, llamada FSM secundaria Nº3. Esta máquina se explica detalladamente más adelante y se muestra en la Figura 15. Por último, una vez que se ha hundido una flota entera de cualquiera de los dos jugadores, el sistema cambia al estado *Fin.* En el cual, se declara que la partida ha finalizado y se muestra por pantalla qué jugador ha resultado vencedor.

A continuación se detalla el funcionamiento de las tres máquinas secundarias de este bloque.

**FSM secundaria Nº1:** Quizás sea la FSM más compleja de explicar. Por ello, se va a dar previamente una visión global de la máquina para intentar comprender claramente el significado de los estados. Cada estado simula la posición de la flecha que apunta a cada barco de la flota. Hay 3 tipos de barcos, cada uno en una fila distinta y el número asociado al nombre del estado representa las casillas que forman cada barco. Se parte del estado *Barco4*, lo que significa que la flecha aparece en la posición que apunta al barco grande. Se puede elegir otro barco mediante los cursores arriba y abajo. Una vez se ha elegido el barco que se desea colocar, se presiona *enter* para seleccionarlo. El sistema se saldría de la FSM secundaria Nº1 y pasaría al estado *Colocar* de la FSM principal. La complejidad del diseño de esta máquina viene dada por que a medida que se van colocando los barcos, y sus respectivos contadores se ponen a ‘0’, el sistema detecta que ya no se puede seleccionar ese tipo de barco y se salta esa opción. Por ejemplo, en el caso de colocar todos los barcos medianos, se salta del barco grande a los pequeños y así con todas las combinaciones posibles. Para intentar aclararlo, en la Figura 13 no se muestran las transiciones para cambiar de un estado a otro, sino que se explican textualmente.

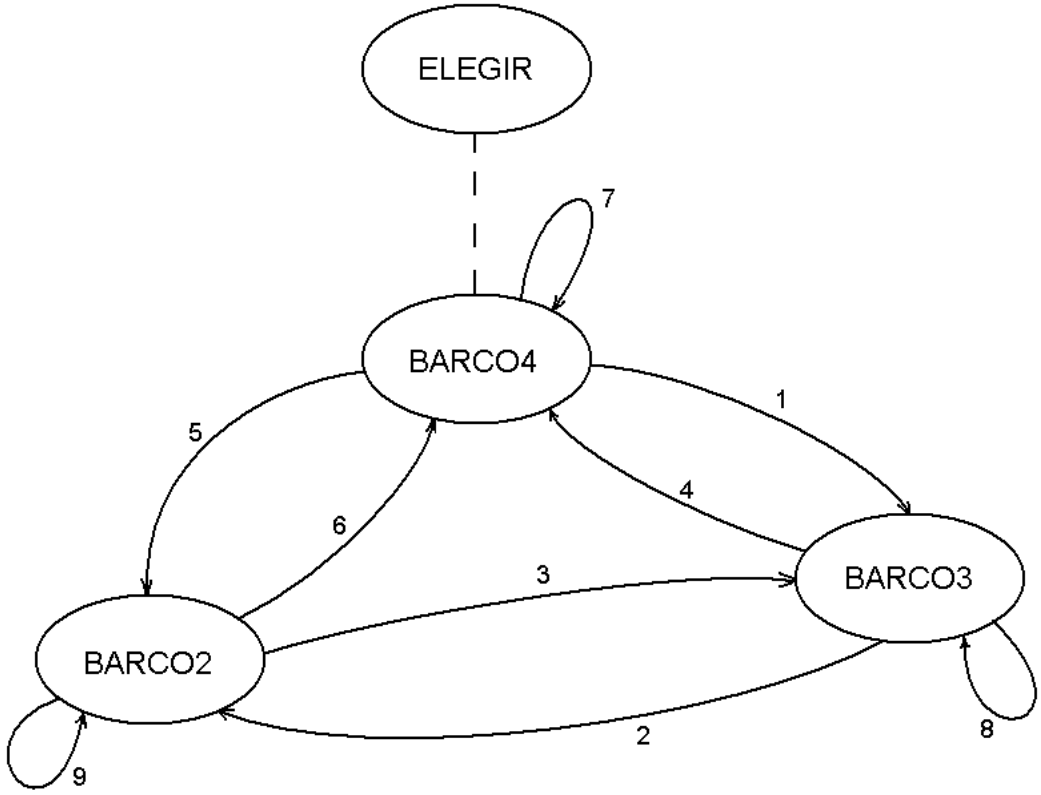



Figura 13. Máquina de estados secundaria Nº1.

1) Mover el cursor hacia abajo o directamente si el barco4 está colocado.
2) Mover el cursor hacia abajo.
3) Mover el cursor hacia arriba.
4) Mover el cursor hacia arriba.
5) Mover el cursor hacia abajo y que todos los barco3 estén colocados.
6) Mover el cursor hacia arriba y que todos los barco3 estén colocados.
7) Mover el cursor hacia arriba o también mover el cursor hacia abajo con los barco3 y barco2 colocados.
8) Mover el cursor hacia arriba o hacia abajo con los barco4 y barco2 colocados.
9) Mover el cursor hacia abajo o mover el cursor hacia arriba con los barco3 y barco4 colocados.

**FSM secundaria Nº2:** Es una máquina de estados que tiene la función de colocar los barcos en el tablero. Se muestra en la Figura 14.

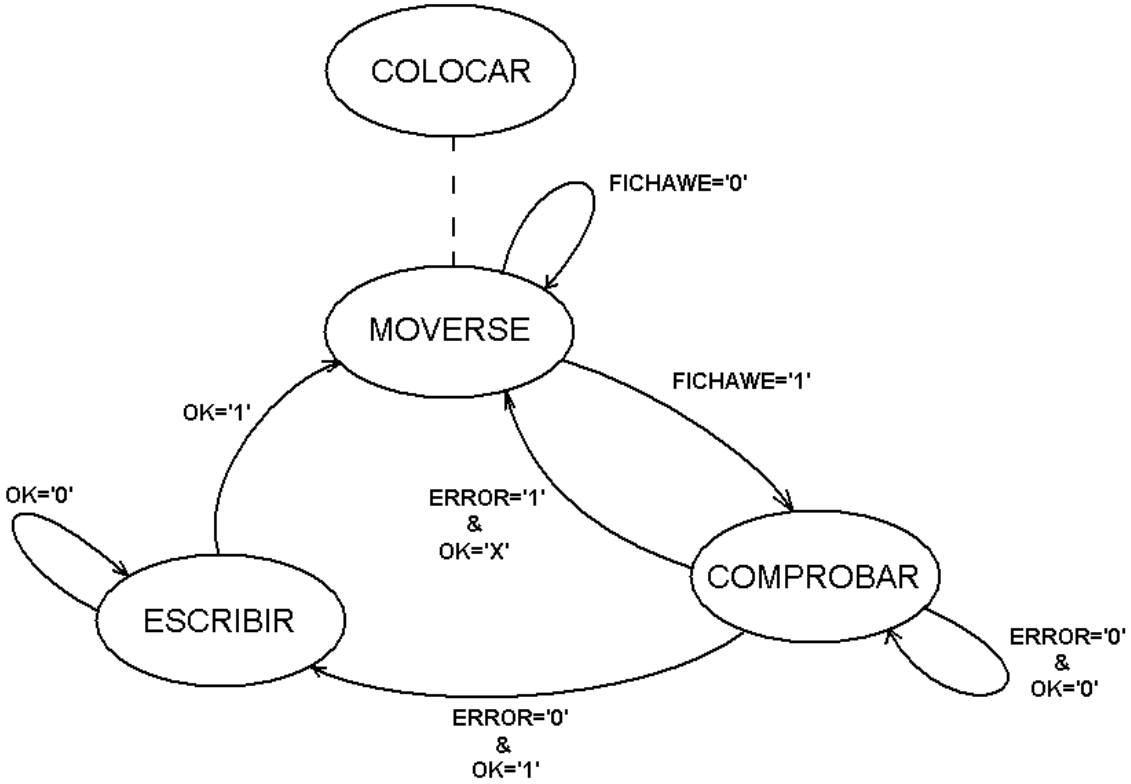



Figura 14. Máquina de estados secundaria Nº2.

Se parte de un estado llamado *Moverse,* en el cual el jugador se moverá por todo el tablero hasta que decida donde colocar el barco, y pulse, *enter*. En ese momento, la máquina se pasa al estado *Comprobar.* El sistema va a buscar dentro de la memoria RAM si la casilla que se ha seleccionado está ocupada por un barco. En caso contrario, continuará comprobando las casillas restantes del barco, en función de la longitud de éste y de la posición en el tablero. Si algunas de las casillas comprobadas están ocupadas o se detecta que el barco se saldría del tablero, entonces no lo colocaría y el jugador volvería a elegir una casilla donde colocarlo de nuevo. Una vez lo coloque y compruebe que se puede situar, la máquina cambia al estado *Escribir.* En este estado, se va a introducir en la memoria RAM los datos del barco colocado. De igual modo que al comprobar si esas casillas están vacías, el sistema va a ir de casilla en casilla escribiendo los datos en la memoria. De esta manera, el sistema ya sabe que en esas posiciones hay un barco, y lo tendrá en cuenta para mostrarlo por pantalla y para no sobrescribirlo en la siguiente colocación de otro barco.

**FSM secundaria Nº3:** Es la máquina de estados correspondiente a la parte del juego donde ambos jugadores intentan hundir la flota del enemigo. Cada jugador dispone de un lanzamiento por cada turno, y los turnos se van alternando hasta proclamarse un campeón. La máquina se muestra en la Figura 15.

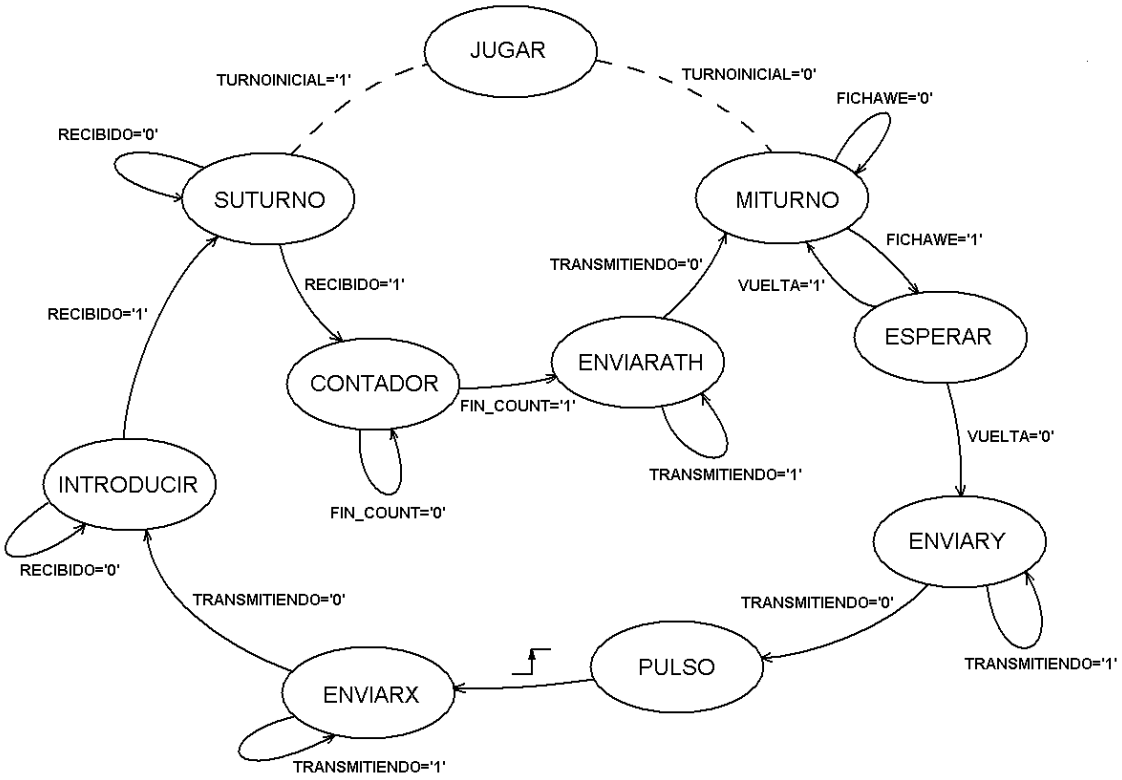



Figura 15. Máquina de estados secundaria Nº3.

Se ramifica del estado *Jugar* de la FSM principal. Cuando se accede a esta máquina, hay dos posibles caminos a seguir, que dependerán de si se posee el primer turno o si éste es del oponente. En el primero de los casos, la máquina pasa al estado *MiTurno*. En éste, el jugador podrá moverse por todo el tablero de la derecha (correspondiente al mapa del jugador oponente) para seleccionar la casilla en la que desee lanzar una bomba. Una vez elegida la posición donde se quiere depositar la bomba, se cambia al estado *Esperar* y se comprueba si en esa casilla ya existe un proyectil. Para realizar esta comprobación se lee la memoria RAM1. Si ya existe una bomba, el sistema retorna al estado anterior para que se vuelva a elegir otra casilla, o en caso contrario, cambia al estado *EnviarY*. Al seleccionar una casilla, se están seleccionando unas coordenadas, X e Y. Son las coordenadas que se van a enviar al oponente. Como indica el nombre del estado, se envía primero la coordenada Y. A continuación, se envía la coordenada X. Hay una separación de un

pulso de reloj entre la transmisión de ambas señales, para sincronizarlas con el receptor del oponente. Por ello, el sistema pasa del estado *EnviarY* al estado *Pulso*, y tras el ciclo de reloj al estado *EnviarX*. Una vez enviadas las coordenadas, el sistema se mantiene a la espera de recibir la respuesta del oponente en el estado *Introducir*. Cuando el receptor ha detectado y analizado la transmisión del jugador contrario, la máquina cambia al estado *SuTurno* cediendo el turno al oponente. Este estado marca el otro camino a seguir al entrar en la FSM secundaria Nº3.

Cuando el sistema se encuentra en *SuTurno*, está esperando a que el oponente envíe las coordenadas que indiquen la casilla donde ha colocado la bomba. Una vez recibidas, se pasa al estado *Contador,* donde se comienza una cuenta para esperar un tiempo y responderle automáticamente una vez haya finalizado. Se cambia al estado *EnviarATH* donde se emite una transmisión indicando al oponente si ha acertado en un barco, si le ha hundido o si por el contrario, su lanzamiento ha caído en el agua. Se vuelve a devolver el turno y se produce este bucle de intercambios de turno hasta que el juego finaliza.

Por otro lado, está la entidad, donde aparecen las entradas y salidas del bloque, y se muestra en la Figura 16.

- ENTIDAD:

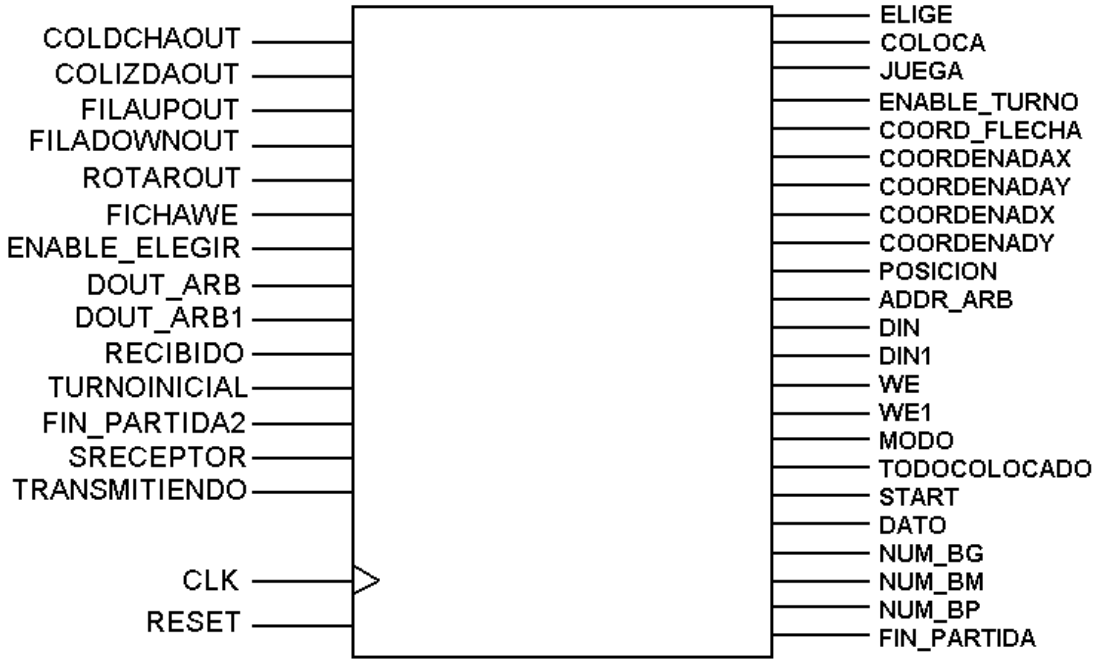



Figura 16. Entidad del Árbitro.

- ENTRADAS:
  - **Clk:** Señal de reloj del sistema. Tiene 50 MHz de frecuencia.
  - **Reset:** Señal de inicialización del sistema. Activa a nivel alto.
  - **ColDchaOut:** Señal que representa el movimiento del cursor a la derecha.
  - **ColIzdaOut:** Señal que representa el movimiento del cursor a la izquierda.
  - **FilaUpOut:** Señal que representa el movimiento del cursor hacia arriba.
  - **FilaDownOut:** Señal que representa el movimiento del cursor hacia abajo.
  - **RotarOut:** Señal que representa la rotación del barco a la hora de colocarlo.
  - **FichaWe:** Señal que representa el *enter* para el sistema.
  - **Enable_elegir:** Señal que pone en funcionamiento el juego.
  - **Dout_arb:** Dato de salida de la memoria RAM que se extrae de una dirección del árbitro.
  - **Dout_arb1:** Dato de salida de la memoria RAM1 que se extrae de una dirección del árbitro.
  - **Recibido:** Activa cuando se ha recibido una trama completa.
  - **TurnoInicial:** Señal que determina cual de los dos jugadores posee el primer turno. Si está activa a nivel alto, el jugador no tiene el turno inicial.
  - **Fin_partida2:** Representa que la partida ha terminado y eres el vencedor.
  - **Sreceptor:** Salida codificada de la trama recibida.
  - **Transmitiendo:** Señal activa mientras se produce una transmisión.
- SALIDAS:
  - **Elige:** Señal que habilita la flecha que apunta a los barcos, para que se pueda elegir el barco a colocar.
  - **Coloca:** Señal que habilita el puntero en el mapa del propio jugador, para que se seleccione la casilla donde colocar el barco seleccionado.
  - **Juega:** Señal que habilita el puntero en el mapa del jugador oponente, para que se seleccione la casilla donde colocar una bomba.

- **Enable_Turno:** Señal activa por interruptor que marca el inicio de una partida.
- **Coord_flecha:** Representa la posición de la flecha dentro del tablero. Tiene 3 posibles posiciones, una por cada tipo de barco.
- **CoordenadaX:** Representa la coordenada X correspondiente al tablero para colocar barcos.
- **CoordenadaY:** Representa la coordenada Y correspondiente al tablero para colocar barcos.
- **CoordenadX:** Representa la coordenada X correspondiente al tablero para colocar bombas.
- **CoordenadY:** Representa la coordenada Y correspondiente al tablero para colocar bombas.
- **Posición:** Representa la posición horizontal o vertical del barco.
- **Addr_arb:** Representa la dirección (X,Y) del árbitro.
- **Din:** Dato de entrada de la memoria RAM que se introduce en una dirección.
- **Din1:** Dato de entrada de la memoria RAM1 que se introduce en una dirección.
- **We:** Habilita la escritura en la memoria RAM.
- **We1:** Habilita la escritura en la memoria RAM1.
- **Modo**: Habilita el receptor para recibir coordenadas o para recibir las posibles contestaciones al envío de una bomba: agua, tocado y hundido (ATH).
- **Todocolodado:** Activa si se han colocado todos los barcos. En función de en que estado de la máquina se valore esta señal, indica cuál de los dos jugadores ha finalizado el primero en colocar los barcos y así poder asignar el turno.
- **Start:** Habilita la transmisión del dato a través de los infrarrojos.
- **Dato:** Trama completa de 14 bits que desea enviarse.

- **Num_bg:** Representa el valor del contador asociado al tipo de barco grande. Su valor de inicio es '1' y se decrementa a medida que se seleccionan.
- **Num_bm:** Representa el valor del contador asociado al tipo de barco mediano. Su valor de inicio es '3' y se decrementa a medida que se seleccionan.
- **Num_bp:** Representa el valor del contador asociado al tipo de barco pequeño. Su valor de inicio es '2' y se decrementa a medida que se seleccionan.
- **Fin_partida:** Indica si el juego ha terminado. Es la concatenación de las señales fin_partida1 (señal que se activa cuando todos tus barcos has sido eliminados, eres el perdedor) y fin_partida2 (señal que envía el otro jugador para indicar que todos sus barcos han sido eliminados y que eres el vencedor).

### 3.2.2 VGA

Se ha diseñado una VGA en función de las especificaciones explicadas en el capítulo anterior. El circuito consta de dos contadores, dos comparadores y un bloque blanqueador. Los contadores generan las coordenadas X e Y, que representan la posición de los píxeles. Cada comparador debe comprobar una coordenada y generar tres señales: la señal de sincronismo (VSync o HSync), una señal que indique el final del ciclo (que se usará para reiniciar el contador correspondiente o para habilitar otro contador), y una señal de blanqueo, que indicará al bloque blanqueador que debe eliminar los colores de las salidas R, G y B. La Figura 17 muestra un esquema aproximado del circuito.

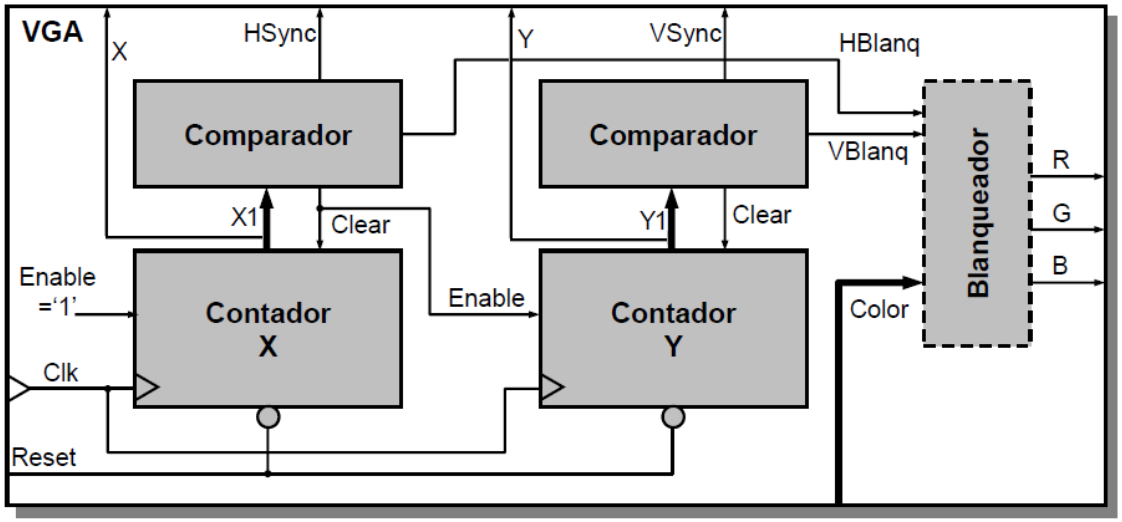



Figura 17. Circuito controlador VGA.

La entidad de la VGA se muestra en la Figura 18.

- ENTIDAD:

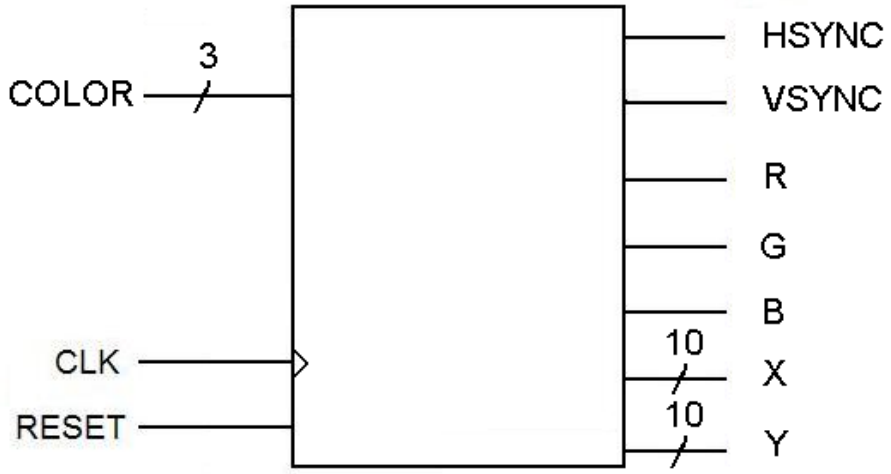



Figura 18. Entidad de la VGA.

- ENTRADAS:
  - **Clk:** Señal de reloj del sistema. Tiene 50 MHz de frecuencia.
  - **Reset:** Señal de inicialización del sistema. Activa a nivel alto.
  - **Color:** Es el código del color que se quiere dar al píxel que se esté procesando en un momento dado. La placa utiliza un convertidor digital analógico de un solo bit por color, por tanto, la señal de Color sólo tendrá 3 bits. Puesto que tenemos 3 bits, el número de colores distintos que podremos representar será $2^3$, es decir 8.
- SALIDAS:
  - **HSync:** Señal de sincronismo horizontal.
  - **VSync:** Señal de sincronismo vertical.
  - **R, G y B:** Son los bits de la señal de Color, pero deberán valer '0' (negro) en los intervalos de sincronismo (entre AH y DH, y entre AV y DV).
  - **X e Y:** Representan las coordenadas del píxel que se está mostrando en cada momento en la pantalla, y coinciden con los valores de los contadores. X es la coordenada horizontal e Y la coordenada vertical.

### 3.2.3 Tablero

El tablero es la interfaz gráfica del juego. Se divide en varias zonas que comprenden las distintas fases a lo largo de la partida.

En primer lugar, hay la zona para elegir barcos, en la cual están el tipo de naves que forman la flota y el número de ellos. Se representa una flecha que apunte a los barcos a modo de puntero. En segundo lugar, está la zona de colocar barcos, que muestra el mapa cuadriculado del propio jugador, donde se sitúa la flota al completo y donde también se verán reflejadas las bombas que envíe el jugador contrario. Cada cuadrícula está designada por unas coordenadas representadas por números (1-8) y letras (A-H). Por otro lado, se representa un cuadrado que cambia de color (verde=tu turno, rojo=turno del oponente) para reflejar el cambio de turno. Además, está la zona donde se lanzan bombas al enemigo, que representa el mapa del jugador oponente, y en la que se muestra si se ha acertado o fallado en el lanzamiento. Por último, cuando se finaliza una partida se muestra por pantalla una felicitación compuesta por un mensaje que indica si se ha ganado o perdido, acompañado de una cara sonriente o triste respectivamente. En la Figura 19 se observa la apariencia del tablero del juego en las distintas fases del juego.

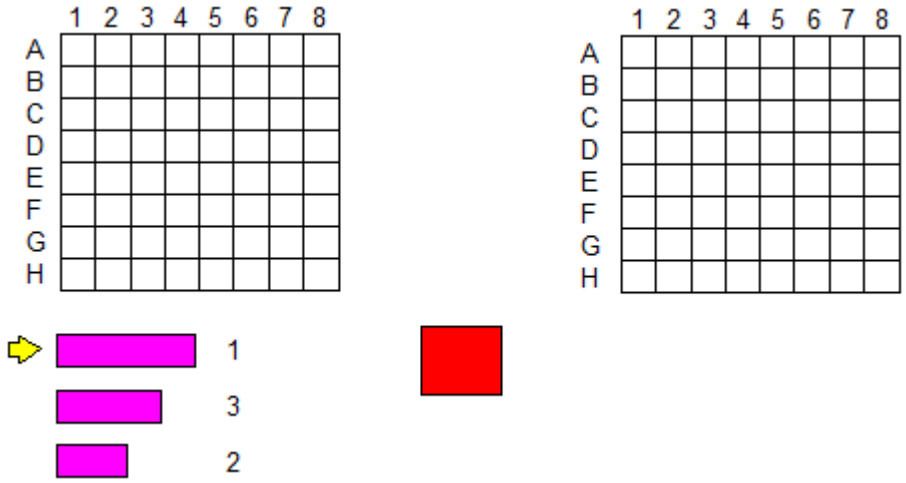



Figura 19. El tablero en las distintas fases del juego.

El diseño del tablero se ha realizado teniendo en cuenta que la resolución de la pantalla es de 640x480. Se han agrupado píxeles en paquetes de 32x32 formando un total de 20x15 cuadrados. La parte gráfica se ha conseguido a través de funciones diseñadas para pintar las letras, los números, la cuadrícula, los barcos, las bombas, la flecha, los punteros, etc. definiendo píxel a píxel lo que se desee mostrar en cada uno de ellos. Se comparan las coordenadas continuamente y se van representando en el tablero a través de la VGA, teniendo en cuenta lo que dictan las memorias RAM. A continuación se muestra la entidad del tablero y se detallan sus I/O.

- ENTIDAD:

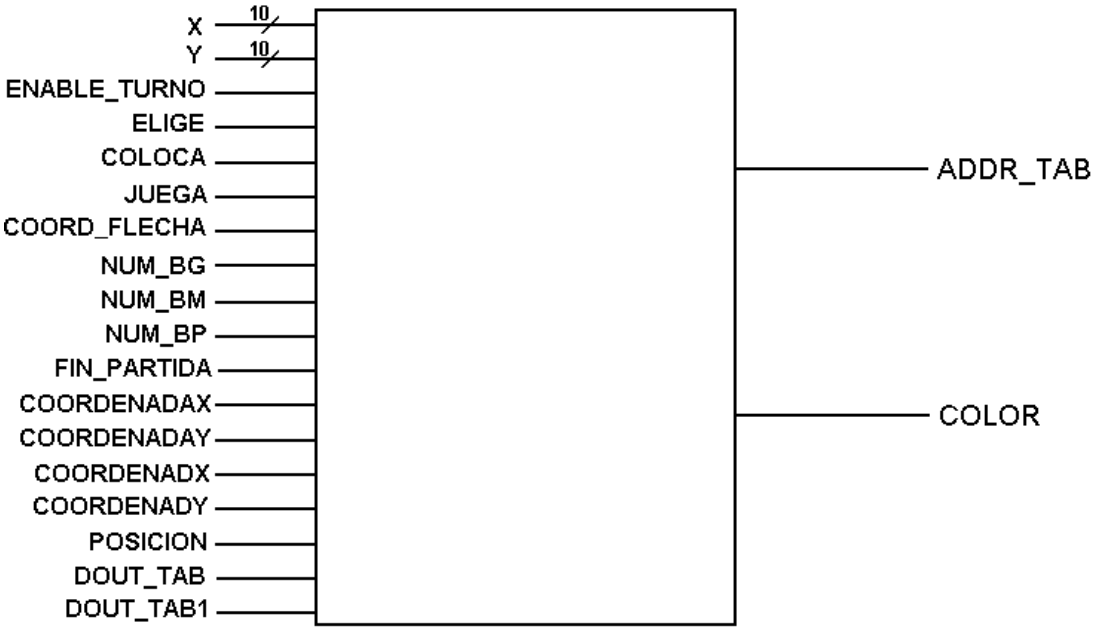



Figura 20. Entidad del tablero.

- ENTRADAS:

  - **X e Y:** Representan las coordenadas del píxel que se está mostrando en cada momento en la pantalla, y coinciden con los valores de los contadores. X es la coordenada horizontal e Y la coordenada vertical.

  - **Enable_turno:** Señal que inicializa el juego.

  - **Elige:** Señal que habilita la flecha que apunta a los barcos, para que se pueda elegir el barco a colocar.

  - **Coloca:** Señal que habilita el puntero en el mapa del propio jugador, para que se seleccione la casilla donde colocar el barco seleccionado.

  - **Juega:** Señal que habilita el puntero en el mapa del jugador oponente, para que se seleccione la casilla donde colocar una bomba.

  - **Coord_flecha:** Representa la posición de la flecha dentro del tablero.

  - **Num_bg:** Representa el valor del contador asociado al tipo de barco grande. Su valor de inicio es '1' y se decrementa a medida que se seleccionan.

- **Num_bm:** Representa el valor del contador asociado al tipo de barco mediano. Su valor de inicio es '3' y se decrementa a medida que se seleccionan.
- **Num_bp:** Representa el valor del contador asociado al tipo de barco pequeño. Su valor de inicio es '2' y se decrementa a medida que se seleccionan.
- **Fin_partida:** Indica si el juego ha terminado. Es la concatenación de las señales fin_partida1 (señal que se activa cuando todos tus barcos has sido eliminados, eres el perdedor) y fin_partida2 (señal que envía el otro jugador para indicar que todos sus barcos han sido eliminados y que eres el vencedor).
- **CoordenadaX:** Representa la coordenada X correspondiente al tablero para colocar barcos.
- **CoordenadaY:** Representa la coordenada Y correspondiente al tablero para colocar barcos.
- **CoordenadX:** Representa la coordenada X correspondiente al tablero para colocar bombas.
- **CoordenadY:** Representa la coordenada Y correspondiente al tablero para colocar bombas.
- **Posición:** Representa la posición horizontal o vertical del barco.
- **Dout_tab:** Dato de salida de la memoria RAM que se extrae de una dirección del tablero.
- **Dout_tab1:** Dato de salida de la memoria RAM1 que se extrae de una dirección del tablero.

▪ SALIDAS:

- **Addr_arb:** Representa la dirección (X, Y) del árbitro.
- **Color:** Es el código del color que se quiere dar al píxel que se esté procesando en un momento dado. La placa utiliza un convertidor digital analógico de un solo bit por color, por tanto, la señal de Color sólo tendrá 3 bits. Puesto que tenemos 3 bits, el número de colores distintos que podremos representar será $2^3$, es decir 8.

### 3.2.4 Emisor de infrarrojos

El emisor de infrarrojos ha sido diseñado para simular el protocolo de transmisión RC5 de Philips. Como ya se ha explicado anteriormente, un bit puede tener dos posibles valores lógicos, 0 o 1. Dentro de cada bit se produce una transición en el medio, por lo que se alterna de nivel bajo a nivel alto o al contrario, según el valor del bit.

Desde el punto de vista del emisor de infrarrojos, la parte a nivel alto de un bit está compuesta por un tren de pulsos. Todos los pulsos tienen la misma forma y periodo, y forman un total de 32 pulsos. El periodo de repetición de la portadora (36KHz) es 27,778us y dado que cada periodo tiene un factor de trabajo del 25%, la duración de cada pulso a nivel alto es 6.944us. Por otro lado, está la parte a nivel bajo de un bit, cuya valor es 0 y tiene la misma duración que la parte alta. Por lo tanto, el periodo de un bit es (32 x 27.778us) x 2 = 1.778 ms. En la Figura 21 se observa la forma de onda que envía el emisor para una mayor comprensión. Ver referencia [5].

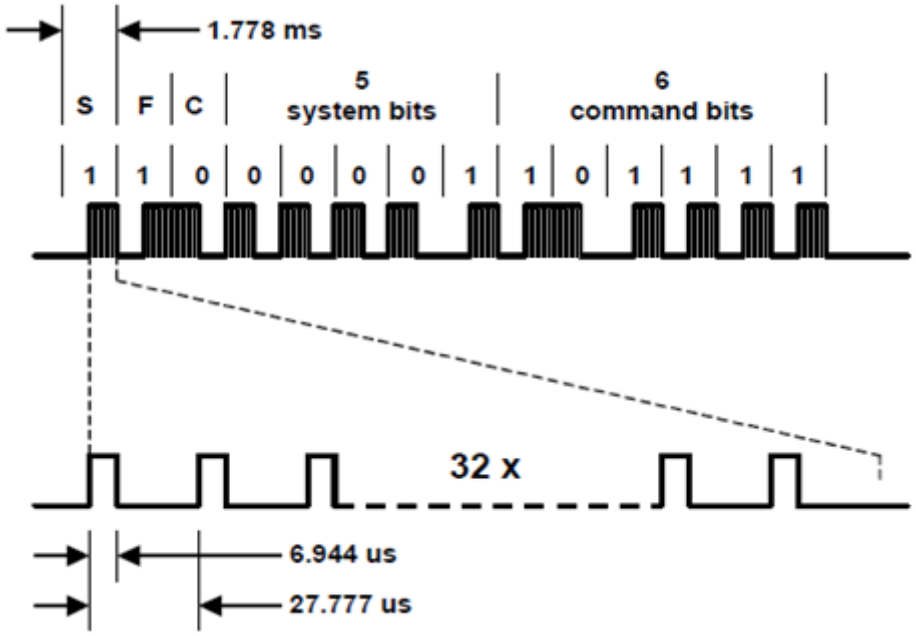



Figura 21. Forma de onda del emisor IR[5].

La transmisión de datos por infrarrojos se hace generalmente modulando en BPSK (Transmisión por Desplazamiento de Fase Binaria) una portadora entre 30 y 60KHz. Para cumplir una especificación del protocolo utilizado, se ha aplicado una señal cuadrada de 36KHz a la base de un transistor tal como muestra la Figura 22, y se ha utilizado un LED de radiación infrarroja incluido en un circuito de

acondicionamiento. Así se consigue que conduzca el LED a la misma frecuencia, emitiendo radiación IR mientras el transistor conduce. Ver referencia [6]

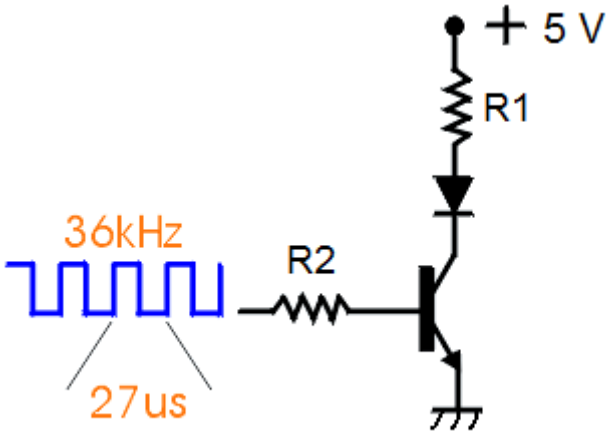



Figura 22. Circuito acondicionador del emisor IR[6].

Se han estudiado varios modelos de emisores infrarrojos y se ha elegido uno acorde a las necesidades. El IR LED debe cumplir con las siguientes características: longitud de onda (λ=950nm), distancia de transmisión (10m), frecuencia a transmitir (36KHz) y tensión (5V) e intensidad (máx.100mA) que soporta. Otra razón a tener en cuenta en la elección ha sido el precio del dispositivo. Finalmente, el modelo utilizado para el emisor de infrarrojos es el SFH-415[7]. Para el transistor se ha utilizado el modelo BC547[8].

La intensidad máxima que debe circular por el IR LED es 100mA. Se han realizado unos cálculos para determinar el valor de las resistencias del circuito. En la práctica, se van a emplear resistencias de ½ w de potencia, para evitar sobrecalentar y dañar el circuito por el paso de una corriente tan alta. Los cálculos son los siguientes:

Datos:

Hfe = β = 300

Vbe = 0.9 V

Vce(sat) = 0.22 V

Ic = 100 mA

Ib = Ic/β = 333 uA

- Vcc = Ic * R1 + Vdiodo + Vce(sat)
- 5 V = 100 mA * R1 + 1.5 V + 0.22 V =>**R1 = 32.8 Ω**
- Vcc = Ib * R2 + Vbe
- 5 V = 333 uA * R2 + 0.9 V => **R2 = 12.3 kΩ**

R1 y R2 representan los valores mínimos para conseguir una corriente en el circuito de 100mA. Como no va a ser necesario utilizar al máximo la potencia del emisor, se han colocado en el circuito los siguientes valores: **R1=470 Ω (1/2 W)** y **R2=10 KΩ (1/2 W)**, permitiendo el paso de una corriente de **6,98 mA**.

El emisor IR diseñado para generar la onda que se aplica en la base del transistor del circuito mencionado anteriormente tiene una entidad. Ésta se muestra a continuación en la Figura 23.

- ENTIDAD:

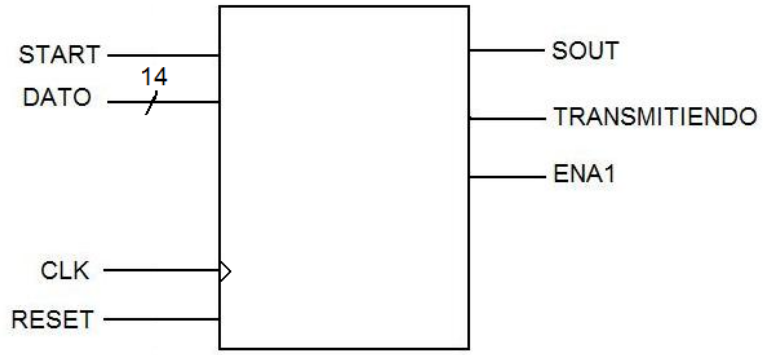



Figura 23. Entidad del emisor de infrarrojos.

- ENTRADAS:

- **Clk:** Señal de reloj del sistema. Tiene 50 MHz de frecuencia.
- **Reset:** Señal de inicialización del sistema. Activa a nivel alto.
- **Start:** Habilita la transmisión del dato a través de los infrarrojos.
- **Dato:** Trama completa de 14 bits que desea enviarse.

- SALIDAS:

- **Sout:** Señal de salida enviada que representa el dato en forma de onda.
- **Transmitiendo:** Señal activa mientras se produce una transmisión.
- **Ena1:** Deshabilita el receptor IR del propio sistema cuando el emisor está enviando, para evitar posibles errores de interpretación a la hora de la práctica.

Para este diseño se ha implementado una máquina de estados, la cual lleva el control casi por completo del funcionamiento de este circuito. Se ha diseñado con el objetivo de generar los trenes de pulsos incluidos en cada bit, y en general, de

conseguir emular la forma de onda emitida. En la Figura 24 se puede observar el diagrama de estados de la máquina y las transiciones de un estado a otro.

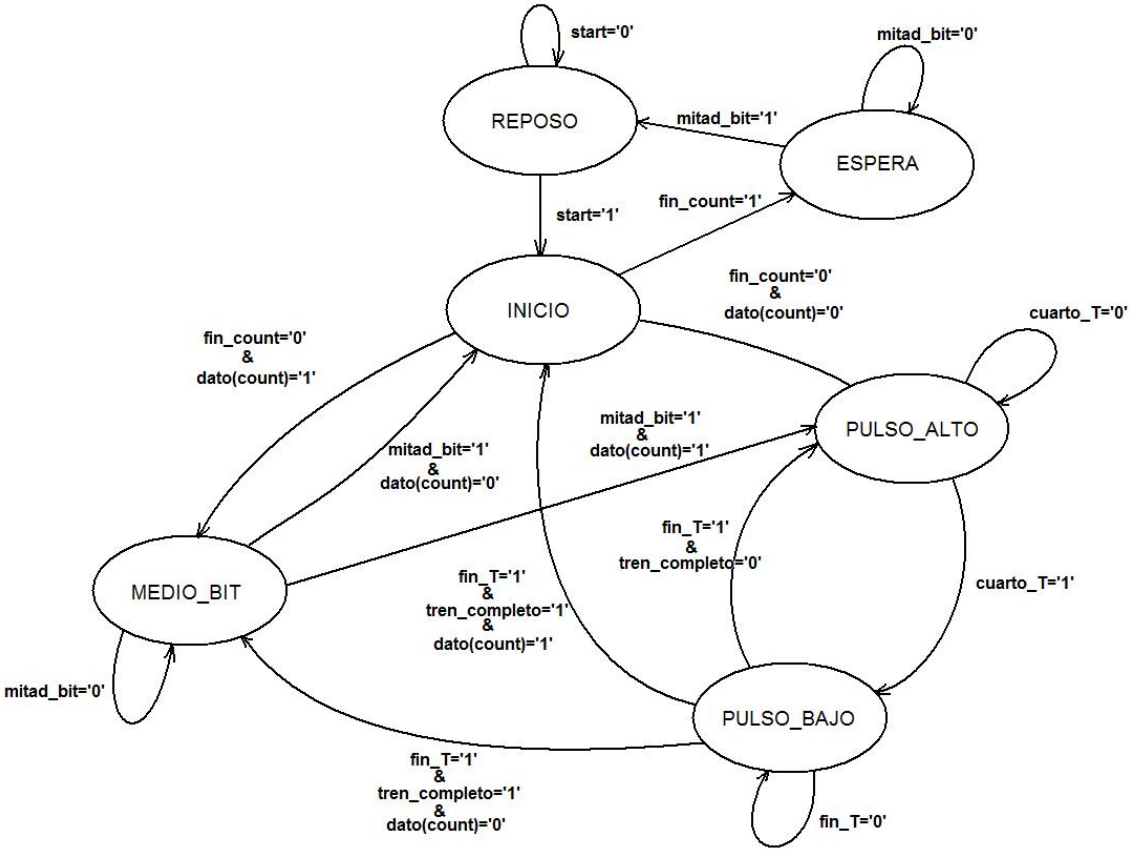



Figura 24. Máquina de estados del emisor de infrarrojos.

La máquina parte de un estado de *Reposo* en el cuál se encuentra siempre que no se envía ninguna transmisión. Las posibles transmisiones enviadas por el emisor en este diseño, pueden tener lugar en cuatro ocasiones:

- Cuando se trata del envío de coordenadas al jugador contrario para indicarle la posición en la cual se ha colocado una bomba.
- Cuando en el caso contrario, se han recibido unas coordenadas y se envía agua, tocado o hundido como respuesta.
- Cuando se le indica al jugador oponente que posee el turno, ya que ha finalizado el primero en colocar todos sus barcos.
- Cuando se le indica al oponente que el juego se ha terminado.

El árbitro del sistema indica mediante la señal de *start* el comienzo de una transmisión, conociendo ya el dato que se quiere transmitir. El sistema cambia a un estado de *Inicio*, donde se comprueba el valor que tiene el primer bit de la trama y dependiendo de si es un 1 o un 0, se desvía por un camino o por otro. Para el caso de que sea un 1, la primera mitad del bit va a estar a nivel bajo, por lo que el sistema cambia a un estado *Medio_bit* donde se activa un contador. Este contador lleva la cuenta de 44450 ciclos de reloj, los cuáles forman medio bit y cuya salida *sout* está a nivel bajo. Una vez finalizada la cuenta, el sistema cambia a la parte de la máquina de estados donde se realiza el tren de pulsos. Esta parte está formada por dos estados, *Pulso_Alto*, con salida *sout* a nivel alto, y *Pulso_Bajo*, con salida *sout* a nivel bajo. En el estado *Pulso_Alto* se va a generar la parte a nivel alto de un pulso. Como ya se conoce, esta parte forma el 25% del pulso, por lo que se va a activar el mismo contador que en el estado *Medio_bit*, y cuando la cuenta llegue a 347 ciclos de reloj, se salta al estado *Pulso_Bajo* que va a generar la parte a nivel bajo de ese pulso. Se continúa con la cuenta y cuando se llega a 1389 ciclos, se consigue haber generado el primer pulso del tren. En ese instante se activa otro contador que lleva la cuenta de los pulsos que se van generando en el tren, y se incrementa a medida que se van generando los pulsos de la manera explicada. Una vez que se han generado los 32 pulsos y por lo tanto, se ha generado el tren completo, el sistema vuelve al estado *Inicio*.

Se vuelve a comprobar qué valor tiene el siguiente bit, incrementando un nuevo contador que refleja el bit dentro de la trama que se está comprobando. Si en este caso es un 0, recorre la máquina por orden inverso a lo explicado anteriormente, es decir, de *Inicio* cambia al estado *Pulso_Alto* y genera el tren de pulsos completo, y después cambiaría al estado *Medio_bit*. El sistema está previsto de un estado determinado *Espera* al cuál se accede una vez se ha generado la onda por completo. En *Espera* se activa el contador de medio bit y una vez que llega al valor de su cuenta máxima, el sistema vuelve a *Reposo*. Este estado es útil y necesario para los casos en los que se emiten dos transmisiones consecutivas, como es el caso del envío de coordenadas, y tiene la función de mantener durante un determinado tiempo al sistema en espera para sincronizar correctamente al emisor con el receptor.

### 3.2.5 Receptor de infrarrojos

La forma de onda de un bit cambia desde el punto de vista de un receptor IR. Como se ha visto anteriormente en el emisor IR, la parte alta de un bit es un tren de 32 pulsos, en cambio el receptor no muestra a su salida estos pulsos ya que la salida está demodulada. Esta demodulación se realiza a través del circuito interno que posee el dispositivo receptor.

En la Figura 25 se observa la forma de onda de un bit a la salida del receptor.


Figura 25. Forma de onda de un bit demodulado.

Cabe destacar que en la práctica el estado de reposo de la señal recibida está a nivel alto, y no a nivel bajo como se observo en el apartado del emisor de IR.

Este diseño se ha realizado como si su nivel de reposo estuviera a nivel bajo, y se ha colocado un inversor a la entrada del circuito para que a la hora de la práctica el diseño funcione correctamente. En la Figura 26 se comparan ambas ondas, tanto la emitida como la recibida.

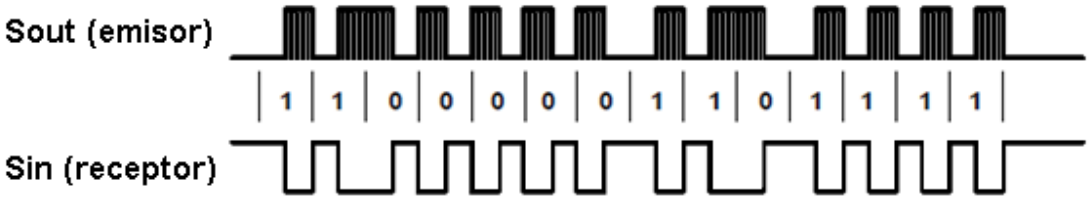



Figura 26. Onda emitida por el emisor y recibida por el receptor una vez demodulada.

Para este diseño se ha utilizado un circuito que controla el receptor de infrarrojos que el autor del proyecto creó para un trabajo dirigido, pero adaptando el diseño a este sistema. El dispositivo infrarrojo es un dispositivo SFH 5510-36 [9]. Tiene las siguientes características: longitud de onda ($\lambda$=940nm), distancia de percepción (30m), frecuencia de la portadora (36KHz), tensión (5V).

Para el correcto funcionamiento del dispositivo se ha montado un circuito de acondicionamiento. Consiste en un filtro paso-bajo conectado al receptor IR. Los valores tanto del condensador como de la resistencia son los mostrados en la Figura 27. Es el circuito recomendado en su hoja de características y por lo tanto, la siguiente figura se ha extraído de tal documento (eliminando una resistencia opcional y un microcontrolador innecesarios).

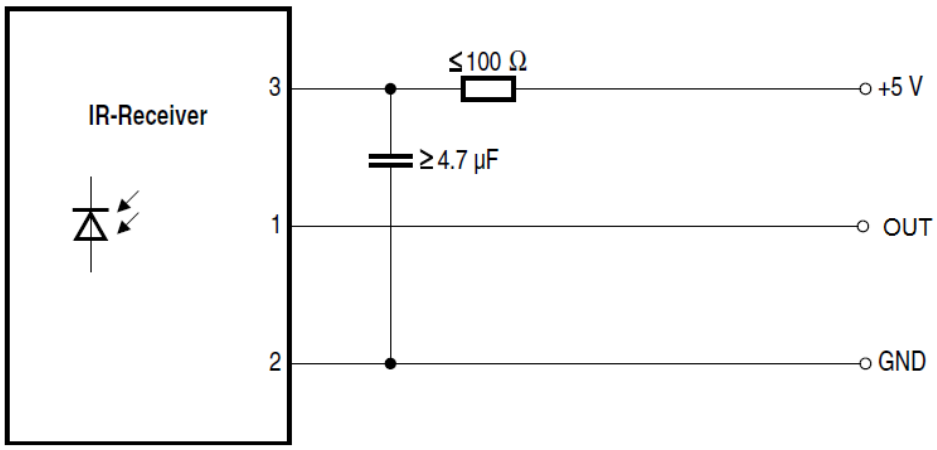



Figura 27. Circuito acondicionador del receptor IR.

La estructura de este diseño está dividida en dos partes: la primera corresponde a la interpretación de los datos recibidos y a su almacenamiento en un registro de desplazamiento. La segunda es la encargada de la decodificación de los datos almacenados y de interpretarlos en función de la fase del juego en la que se encuentre. El conjunto de las dos forman el circuito global del receptor.

El diagrama de bloques correspondiente a la primera parte del diseño se muestra en la Figura 28.

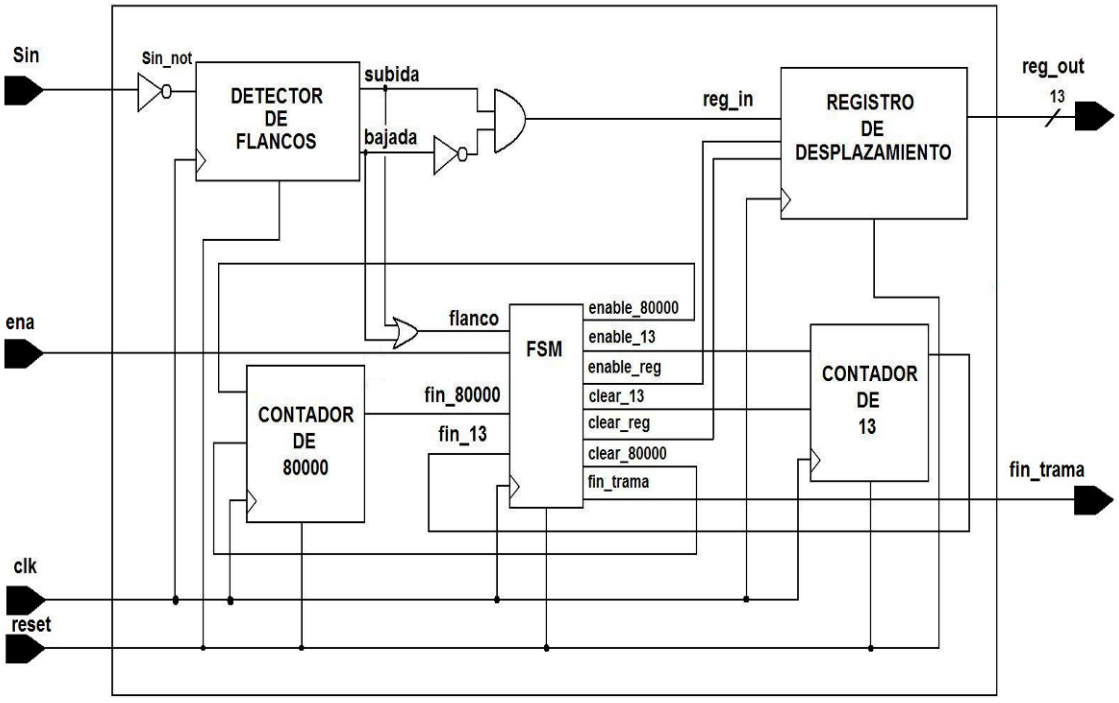



Figura 28. Bloque que interpreta y almacena los datos recibidos en el receptor IR.

El diagrama de bloques del circuito global se muestra en la Figura 29.

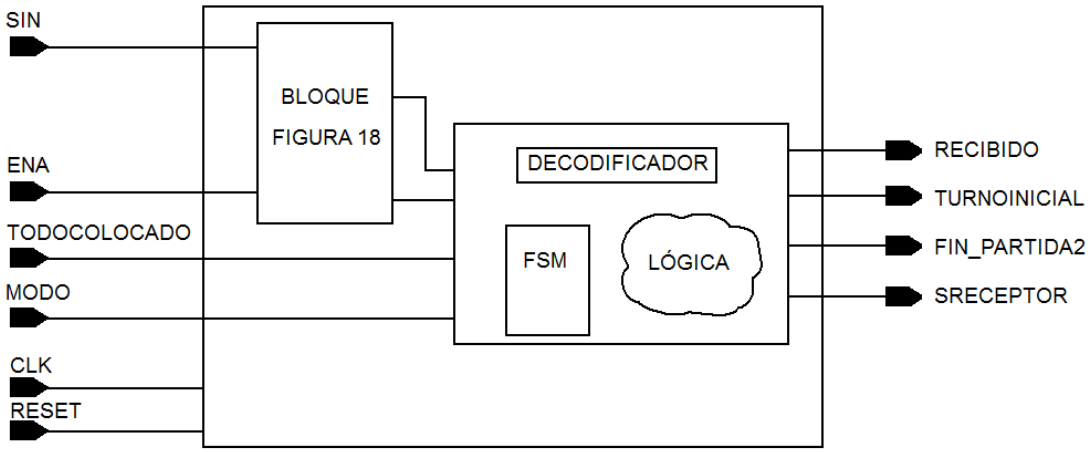



Figura 29. Bloque del circuito global del receptor IR.

Cada parte del diseño tiene una máquina de estados, acompañada de distintos bloques como: contadores, detectores de flanco, decodificadores, etc. A continuación se explica la función que desempeñan en el diseño.

- DETECTOR DE FLANCOS:

La señal de entrada *Sin_not* alterna su valor lógico de nivel alto a bajo o al contrario, por lo que es necesario crear una forma de saber cuando la señal cambia de valor. La solución a este problema la tiene el detector de flancos. Éste indica a la salida cuando ha llegado un flanco de subida, o por el contrario cuando el flanco detectado ha sido de bajada. Su esquema interno se muestra en la Figura 30.

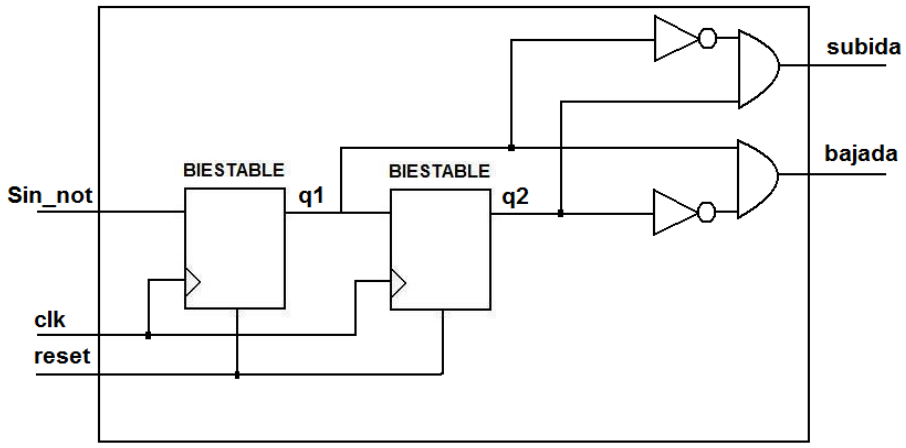



Figura 30. Detector de flancos.

Los biestables registran el valor de la señal entrante en cada flanco de subida del reloj. Cuando se produzca un cambio de nivel en la señal de entrada *Sin_not*, en el siguiente flanco de subida del reloj la señal *q1* habrá obtenido el valor nuevo de la señal entrante, y en el siguiente flanco de subida lo obtendrá la señal *q2*. Por lo que a la salida del detector de flancos se tendrá un '1' en la señal *subida*, en el caso de que *q1* sea '1' y *q2* sea '0'. O por el contrario se tendrá un '1' en la señal *bajada* en el caso de que *q1* sea '0' y *q2* sea '1'. Para los otros dos casos restantes que serían cuando ambas señales intermedias (*q1* y *q2)* tienen valor '0' o '1', querrá decir que no ha habido ningún flanco en la señal de entrada y por lo tanto se tendrá el valor '0' en los dos salidas del detector.

- REGISTRO DE DESPLAZAMIENTO:

Tiene como función registrar todos los bits que se han detectado en el receptor de infrarrojos. Tiene una longitud de 13 bits aunque la trama sea de 14 bits, y esto es debido a que el bit de comienzo o bit *Start* no se almacena.

- CONTADOR DE 80000:

Este contador tiene como función llevar la cuenta de los ciclos de reloj que deben transcurrir, para que el sistema no tenga en cuenta todos los flancos que detecte el detector de flancos, sino tan solo los que son necesarios para el diseño.

- CONTADOR DE 13:

Este contador tiene como función llevar la cuenta de los bits que van incorporándose al registro de desplazamiento.

- DECODIFICADOR:

En este bloque del circuito se decodifica la trama enviada al receptor de infrarrojos. Aplicando una salida a cada entrada recibida.

Por otro lado, se ha implementado la FSM correspondiente al primer bloque. En la Figura 31 se puede observar el diagrama de estados de la máquina y las transiciones de un estado a otro.

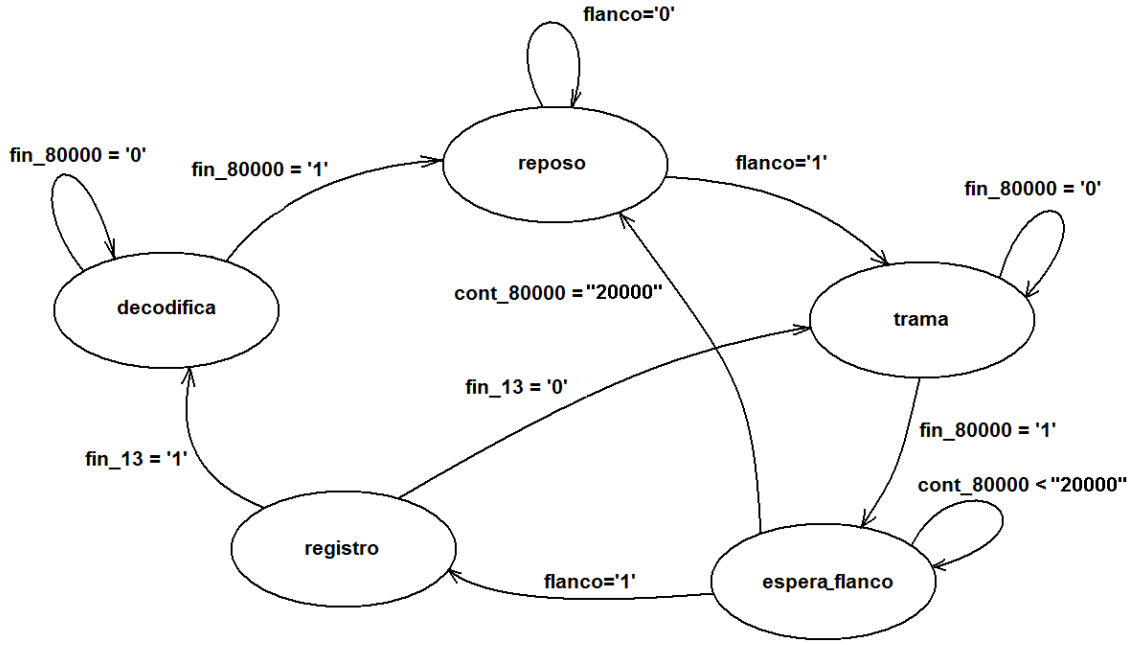



Figura 31. Máquina de estados del receptor de IR para registrar la trama recibida.

El sistema parte de un estado de *Reposo,* en el cual se encuentra siempre que no se le envíe ninguna transmisión al receptor de infrarrojos. El primer flanco que llega será el que determina el bit *start*. El sistema cambia a *Trama* y en este estado se pone en marcha un contador que contará hasta 80000 ciclos de reloj, para poder salvar el flanco intermedio de cada bit que no tiene trascendencia para el diseño. En la Figura 32 se aclara gráficamente esta explicación.

$$Tclk = \frac{1}{Fclk} = \frac{1}{50\ MHz} = 20\ ns$$

1 bit = 1,778 ms -> 88900 ciclos de reloj

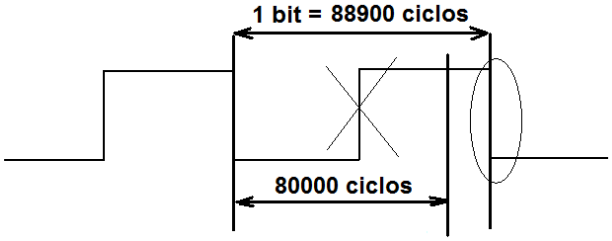



Figura 32. Salto del flanco no detectable.

Una vez ha finalizado la cuenta el sistema pasa al estado *Espera_flanco*. Permanece en ese estado hasta que el detector de flancos le envía la señal correspondiente indicando que se ha detectado un flanco. En este estado también se activa el contador de 80000, y su explicación se dará con mayor detalle en el capítulo de resultados prácticos. Para una mejor comprensión se adelanta que se ha colocado para evitar fallos a la hora de la práctica. En el caso de superar la cuenta de 20000 ciclos, el sistema regresa al estado *Reposo* a esperar una nueva transición.

Como ya se tiene constancia de que un flanco ha llegado (si es de bajada o es de subida) se conoce el valor lógico del bit que ha llegado, por lo que este bit se almacena en un registro de desplazamiento. El bit se almacena en el estado *Registro* de la FSM. En la transición del cambio de estado de *Espera_flanco* a *Registro*, se activa otro segundo contador que lleva la cuenta de los bits que entran en el registro de desplazamiento. Si este contador ha llegado a la cuenta de 13 bits, salta al estado *Decodifica* del sistema. En el cuál se activa una señal llamada *fin_trama* que servirá de habilitador para registrar la salida final del circuito. En el caso de que la cuenta no llegue a 13 bits, significa que la trama no está completa todavía y desde *Registro* se salta a *Trama,* volviendo a realizar lo comentado anteriormente. Por último, una vez se ha recogido la palabra enviada en el registro de desplazamiento tan solo queda decodificar e interpretar esa trama.

En este punto del sistema, es necesario poner en marcha el decodificador. Entrando en funcionamiento la segunda parte del diseño del receptor. Y por lo tanto, entrando en funcionamiento la segunda máquina de estados que hay en este bloque. En ella, se va a analizar e interpretar las transmisiones en función de en que fase se encuentre del juego. Las tres posibles transmisiones que le pueden llegar son: una referente al primer turno, las coordenadas de una bomba y le respuesta al lanzamiento de una bomba. En la Figura 33 se muestra la FSM de la segunda parte del receptor IR.

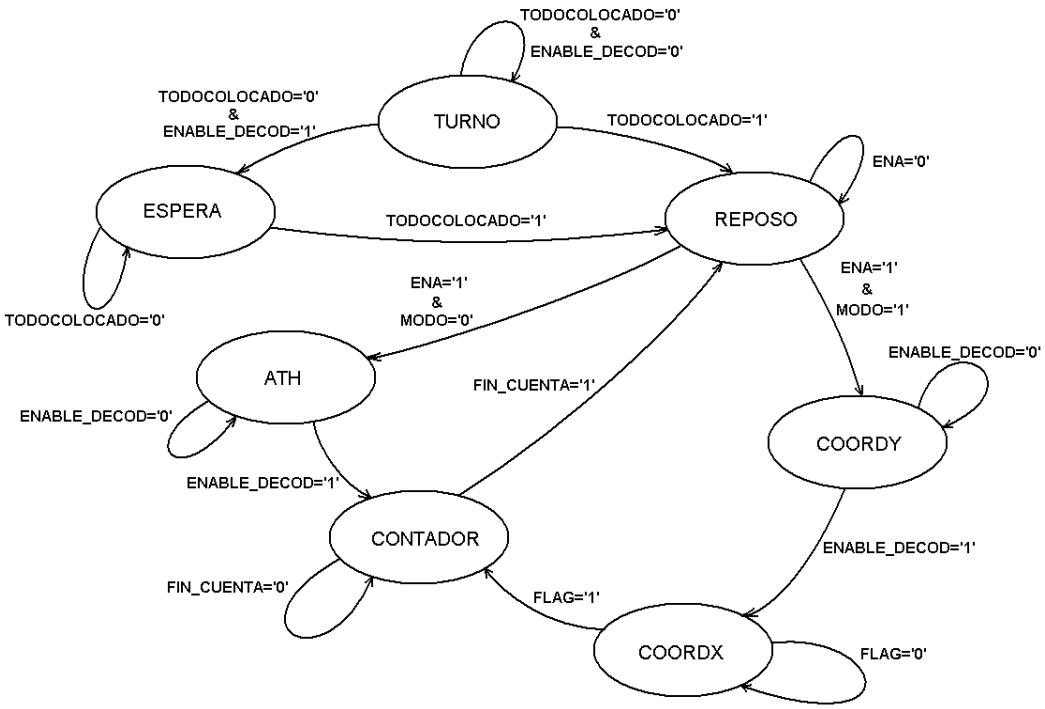



Figura 33. Máquina de estados del receptor IR para interpretar los datos recibidos.

El sistema receptor IR parte de un estado inicial llamado *Turno*. Se mantiene en este estado hasta conocer que jugador ha terminado antes de colocar los barcos en su tablero y así, decidir quien posee el turno inicial. El jugador que primero coloque sus barcos cambia al estado *Reposo,* y le envía al oponente una señal para indicarle que ya ha terminado.

Al recibir esta señal, el jugador contrario pasa al estado *Espera* y se mantiene hasta que termine de colocar toda su flota. En este estado se activa una señal que le deshabilita el turno, y por lo tanto, asigna el primer turno al jugador que primero ha terminado. Una vez que también ha colocado su flota, pasaría al estado *Reposo*. El sistema se encontrará en el estado *Reposo* para ambos jugadores, pero con el turno asignado.

Comienza la partida y el jugador que dispone del turno enviará una bomba. Ambos receptores se encuentran activados, pero para evitar que los dos interpreten siempre las señales que reciban, se activará únicamente el receptor contrario al jugador que envía los datos. O lo que es lo mismo, se deshabilita el receptor del jugador que transmite. Esto se consigue con una señal que proviene del emisor cuando está transmitiendo, y que está conectada a la entrada del receptor del propio jugador. Esta señal se activa cuando se transmite y por lo tanto, permite interpretar la señal que se le envía, mientras el receptor del propio jugador ignoraría la señal que también recibe.

Adaptando esta idea en la FSM, se produciría un cambio de estado u otro en función de la posesión del turno. Por un lado, el jugador que no posee el turno recibe las coordenadas de la bomba enviada (modo=’1’). Cuando se le habilita el receptor, se pasa al estado *CoordY*, en el cual se registra la coordenada que se le ha enviado y se codifica en una señal. A continuación, se cambia al estado *CoordX* donde se registra la coordenada y se codifica en una señal combinada con la anterior. Una vez se tiene esta señal, que será una salida del receptor para que el bloque *árbitro* trabaje con ella, el sistema cambia de estado a *Contador*. En este estado se activa un contador durante un determinado tiempo para sincronizar la comunicación y vuelta al estado de *Reposo,* donde se mantendrá a la espera de recibir más señales infrarrojas.

Por otro lado, el jugador que tiene el turno va a esperar que se le envíe la respuesta al lanzamiento de su bomba, ya sea *agua, tocado o hundido* (modo=’0’). Al activarse su receptor, la máquina pasaría al estado *ATH*. En el cual, una vez recibida la señal y codificada la interpretación de la onda, se pasaría al estado *Contador* para sincronizar la comunicación. Al igual que por el otro camino, una vez finaliza la cuenta del contador, el sistema cambia al estado *Reposo*.

La máquina se bifurcará por un camino o por otro dependiendo del turno y siempre realizará los mismos pasos anteriores por cada senda.

A continuación se muestra la entidad de este bloque en la Figura 34.

- ENTIDAD:

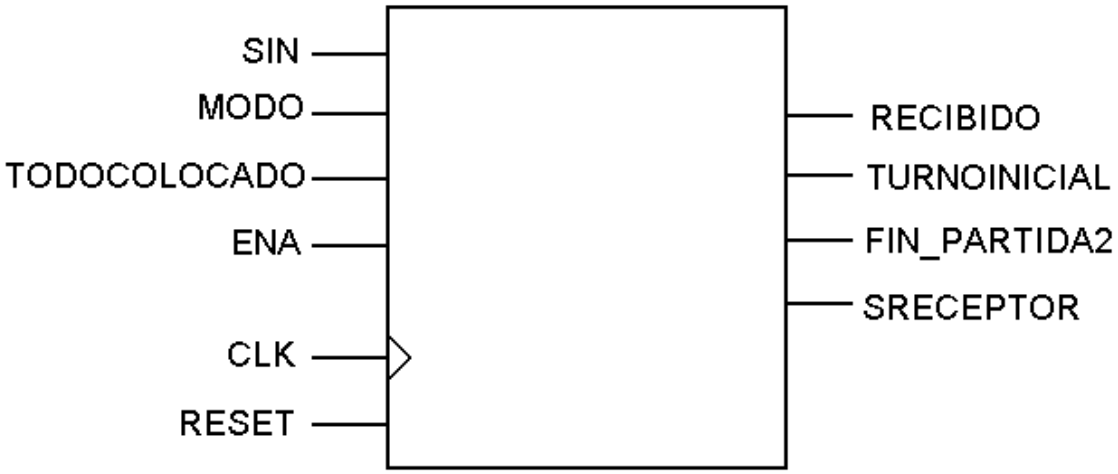



Figura 34. Entidad del receptor IR.

- ENTRADAS:

- **Clk:** Señal de reloj del sistema. Tiene 50 MHz de frecuencia.
- **Reset:** Señal de inicialización del sistema. Activa a nivel alto.
- **Sin:** Señal de entrada recibida por el componente receptor de infrarrojos.
- **Todocolocado:** Señal que se activa si se han colocado todos los barcos. En función de en que estado de la máquina se valore esta señal, indica cuál de los dos jugadores ha finalizado el primero en colocar los barcos y así poder asignar el turno.
- **Modo:** Habilita el receptor para recibir coordenadas o para recibir las posibles contestaciones al envío de una bomba: agua, tocado y hundido (ATH).
- **Ena:** Deshabilita el receptor cuando el emisor envía. A esta entrada se le conecta la salida *ena1* del emisor IR.

- SALIDAS:

- **Recibido:** Activa cuando se ha recibido una trama completa.

- **TurnoInicial:** Indica que jugador tiene el turno para comenzar el juego.
- **Fin_partida2:** Indica si el juego ha terminado.
- **Sreceptor:** Salida codificada de la trama recibida.

### 3.2.6 Memorias RAM

El sistema necesita tener memoria para recoger y almacenar una serie de datos necesarios para reflejar el estado de la partida en el tablero. Se han diseñado dos memorias RAM: Memoria RAM y Memoria RAM1.

En primer lugar, la Memoria RAM se ha diseñado para recoger los datos referentes al mapa del propio jugador. La memoria está formada por 64 filas, que son las direcciones necesarias para completar un tablero de 8x8 casillas (1 fila por cada casilla). Los datos almacenados en cada fila son: el tamaño del barco que el jugador ha colocado, la posición del barco, la zona del barco y la posible colocación de una bomba por parte del jugador contrario. La dirección de cada fila se representa con los 3 últimos bits de los vectores X e Y del tablero, y la memoria tiene dos puertos de acceso, uno para que acceda el árbitro y otro para el bloque tablero.

El tipo de barco puede tener los siguientes valores: un barco grande (variable=grande), 3 barcos medianos (variables=med1/med2/med3), 2 pequeños (variables=peq1/peq2) y la posibilidad de que la casilla esté vacía (variable=vacío). Para el barco existen dos posibles posiciones: horizontal y vertical (variables=hor/ver). Todo barco tiene una zona inicial y final. En el caso del barco pequeño, está compuesto por estas zonas, para el barco mediano se identifican las zonas de inicio, mitad y final. Y el barco grande está formado por una zona de inicio, dos mitades y la final (variables=inicio/mitad/final). Por último, si el oponente ha colocado una bomba en una casilla, la memoria almacenará también este dato. La memoria tiene unos valores por defecto con los que se inicializa.

En segundo lugar, la Memoria RAM1 está diseñada para recoger los datos que el jugador contrario envía al propio jugador, cuando éste envía una bomba, y que se refleja en el mapa del oponente. La memoria RAM1 tiene la misma capacidad que la memoria RAM, ya que cubre las dimensiones para un mismo tablero de 8x8 casillas. En este caso, el tamaño por cada fila viene definido por un tipo enumerado designado como “torpedo”, que recoge los siguientes posibles valores: nobomba, agua, tocado y hundido. Al igual que en la memoria RAM, está inicializada a unos valores por defecto. Para una mayor comprensión del funcionamiento de las memorias, en la Figura 35 se puede observar el contenido de una memoria por defecto al iniciarse y con un valor almacenado.

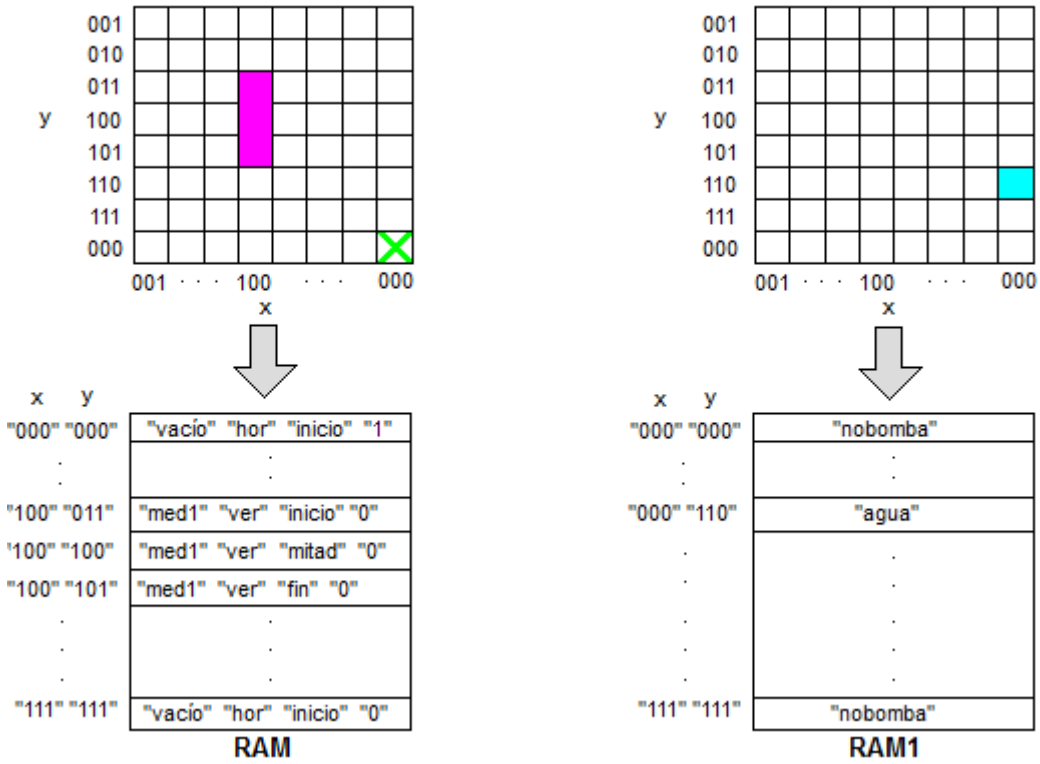



Figura 35. Memorias RAM y RAM1.

El propio jugador tiene colocado un barco mediano y le han mandado una bomba, mientras que el jugador oponente ha devuelto la respuesta como agua, al envío de otra bomba del primer jugador.

A continuación se detallan las entidades de ambas memorias en la Figura 36.

- ENTIDAD RAM:

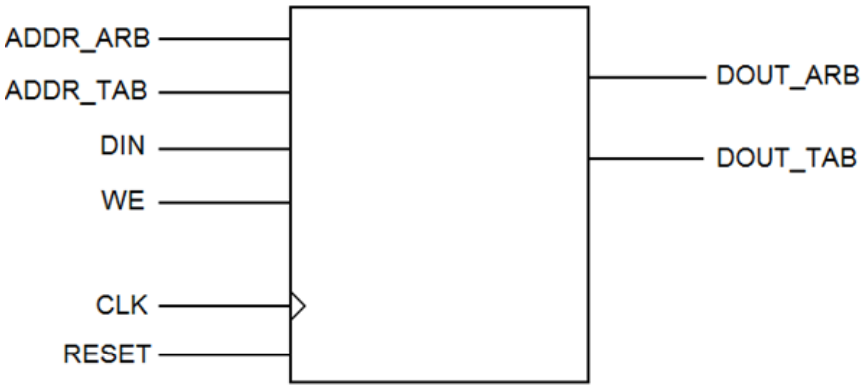



Figura 36. Entidad de la memoria RAM.

- ENTRADAS:

- **Clk:** Señal de reloj del sistema. Tiene 50 MHz de frecuencia.
- **Reset:** Señal de inicialización del sistema. Activa a nivel alto.
- **Addr_arb:** Representa la dirección (X,Y) del árbitro.
- **Addr_tab:** Representa la dirección (X,Y) del tablero.
- **Din:** Dato de entrada de la memoria que se introduce en una dirección.
- **We:** Habilita la escritura en la memoria RAM.

- SALIDAS:

- **Dout_arb**: Dato de salida de la memoria que se extrae de una dirección del árbitro.
- **Dout_tab**: Dato de salida de la memoria que se extrae de una dirección del tablero.

La entidad de la memoria RAM1 es idéntica a la de la memoria RAM, pero diferenciando algunas I/O añadiéndolo un '1' al final de su nombre.

### 3.2.7 Interfaz jugador-computer

Para poder interactuar entre el jugador y el juego se ha diseñado una interfaz capaz de satisfacer los movimientos necesarios para todos los estados del juego. Para ello se dispone de una serie de pulsadores e interruptores incorporados en la FPGA que serán las entradas a la interfaz, y por donde, el jugador tendrá el control total de los movimientos en la partida.

Para diseñar esta interfaz, se ha estudiado previamente cuales son los movimientos que van a hacer falta para interactuar en todos los estados de la partida. Los movimientos necesarios para jugar son los cursores: arriba, abajo, derecha e izquierda, un movimiento de rotación si se desea para girar el barco al colocarlo y un *enter* que tendrá diferentes funciones, dependiendo del estado de la partida. A parte de estos movimientos, los jugadores necesitarán un *reset* para reiniciar el circuito. Por lo tanto, se necesitan 7 pulsadores o interruptores por cada jugador. Estas entradas están asociadas a los movimientos anteriormente mencionados.

Por otro lado, es necesario saber el modo de detectar estas entradas y transformarlas en movimientos. Cada vez que se active uno de los pulsadores se debe detectar el flanco de subida. A parte de esto, se debe tener especial cuidado en

evitar que el diseño detecte los rebotes al soltar el pulsador. Se ha diseñado la máquina de estados mostrada en la Figura 37.

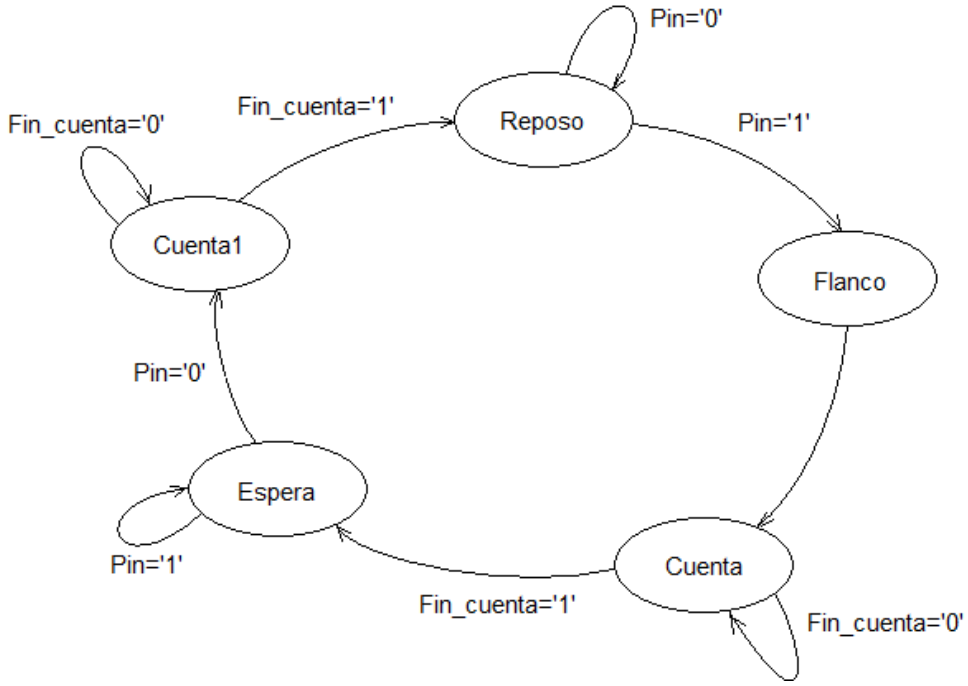



Figura 37. Máquina de estados de la interfaz jugador-computer.

La máquina de estados parte de un estado de *Reposo*, en el cuál el sistema está esperando a que el jugador realice algún movimiento. Si el jugador activa uno de los pulsadores, la entrada correspondiente se pone a '1' durante el tiempo que este pulsando el jugador y cuando deja de pulsar, se producen una serie de rebotes en la onda. Para tratar este tipo de señal cuando el jugador ha pulsado, el sistema detectará el flanco de subida y la máquina pasa a un estado llamado *Flanco*, en el que únicamente permanece un ciclo de reloj. Este estado sirve para habilitar un *enable* asociado al movimiento que se ha ejecutado. Una vez se ha cumplido el ciclo de reloj, se pasa a un estado llamado *Cuenta* donde se activa un contador para mantener el sistema durante un tiempo determinado en ese estado mientras el jugador está pulsando. Cuando se termina el contador, la máquina pasa al estado *Espera*, en el cuál se mantiene esperando a que la señal de entrada baje a nivel '0'. Es en ese momento cuando el sistema cambia de estado a *Cuenta1*, donde se activa de nuevo el contador anterior, con el fin de mantener el sistema en ese estado mientras se producen los rebotes en la señal de entrada e ignorarlos. Con estos estados se pretende que dure lo que dure el accionamiento del pulsador, se salven los rebotes y se interprete un único flanco de subida para cada pulsado. Una vez terminada la cuenta, el sistema vuelve al estado de *Reposo* a esperar otro

movimiento del jugador. A continuación se detalla la entidad mostrada en la Figura 38.

- ENTIDAD:

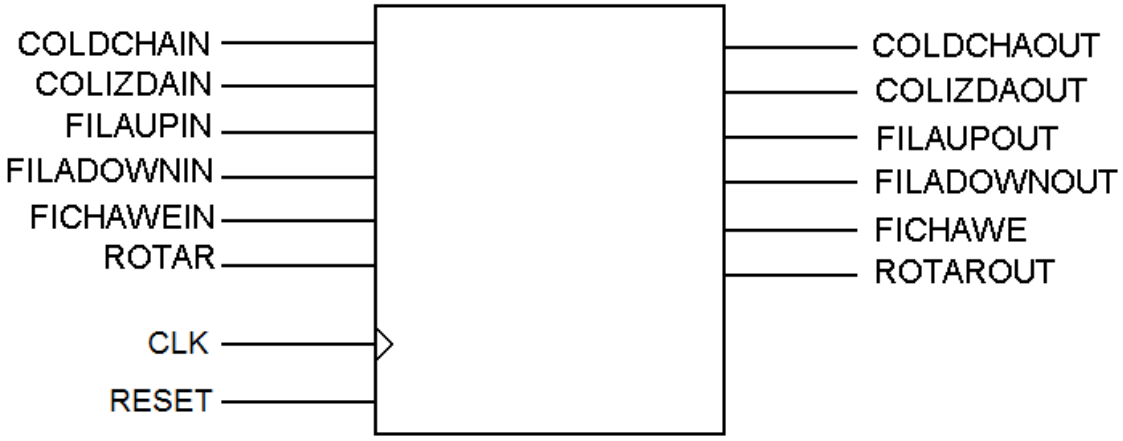



Figura 38. Entidad de la interfaz jugador-computer.

- ENTRADAS:
  - **Clk:** Señal de reloj del sistema. Tiene 50 MHz de frecuencia.
  - **Reset:** Señal de inicialización del sistema. Activa a nivel alto.
  - **ColDchaIn:** Señal que representa el pulsador del cursor a la derecha.
  - **ColIzdaIn:** Señal que representa el pulsador del cursor a la izquierda.
  - **FilaUpIn:** Señal que representa el pulsador del cursor hacia arriba.
  - **FilaDownIn:** Señal que representa el pulsador del cursor hacia abajo.
  - **FichaWeIn:** Señal que representa el pulsador del *enter* para el sistema. Depende del estado del juego tendrá una función u otra. Se utiliza para seleccionar el barco, para colocar el barco elegido y para indicar la casilla donde se desea lanzar una bomba.
  - **Rotar:** Señal que representa el pulsador de la rotación del barco a la hora de colocarlo. Se puede poner horizontal o vertical, es un giro de 90º, no gira sobre sí mismo 360º.
- SALIDAS:
  - **ColDchaOut:** Señal que representa el movimiento del cursor a la derecha.

- **ColIzdaOut:** Señal que representa el movimiento del cursor a la izquierda.
- **FilaUpOut:** Señal que representa el movimiento del cursor hacia arriba.
- **FilaDownOut:** Señal que representa el movimiento del cursor hacia abajo.
- **FichaWe:** Señal que representa el *enter* para el sistema.
- **RotarOut:** Señal que representa la rotación del barco a la hora de colocarlo.

# CAPÍTULO 4

## 4 RESULTADOS Y VALIDACIÓN

Una vez el sistema ha sido diseñado, se han realizado comprobaciones prácticas para demostrar que el diseño funciona correctamente. A lo largo del desarrollo del sistema se han comprobado, con multitud de simulaciones en ModelSim, todas las facetas que tiene que cumplir.

Las prácticas se han realizado con un montaje que simula una partida real. Es un juego para dos jugadores, por lo que se han montado dos circuitos idénticos. Cada circuito está compuesto por: una placa FPGA, un monitor con vga, un circuito receptor y otro emisor de infrarrojos montados en una placa protoboard. Se puede observar en la Figura 39 como queda montado el circuito final.

Figura 39. Montaje para dos jugadores.

La FPGA está conexionada con el monitor mediante el conector vga que incluye. Para la interfaz con el jugador se utilizan los interruptores y el pulsador (para el reset) que se encuentran en la parte inferior de la placa FPGA. Además, el circuito emisor de un jugador se monta frente al circuito receptor del contrario. Ambos en una placa protoboard. Se alimentan a +5V y las masas de todo el circuito están cortocircuitadas entre sí. En la Figura 40 se puede observar el conexionado de los circuitos infrarrojos con los dispositivos controladores.

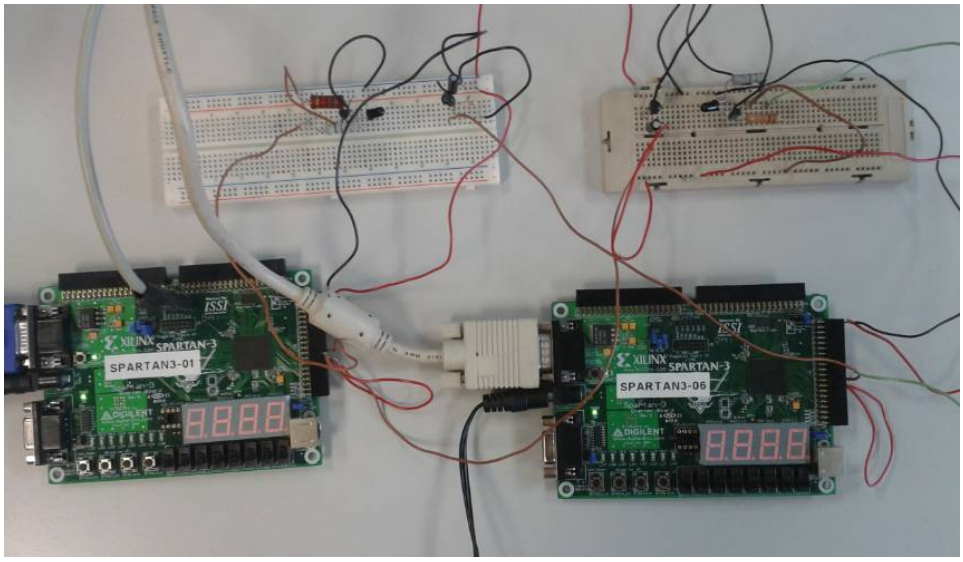



Figura 40. Circuito emisor y receptor conexionado a FPGA.

En la Figura 41 se puede observar con alto detalle los circuitos acondicionadores que se montan en la práctica para emitir o recibir las señales.

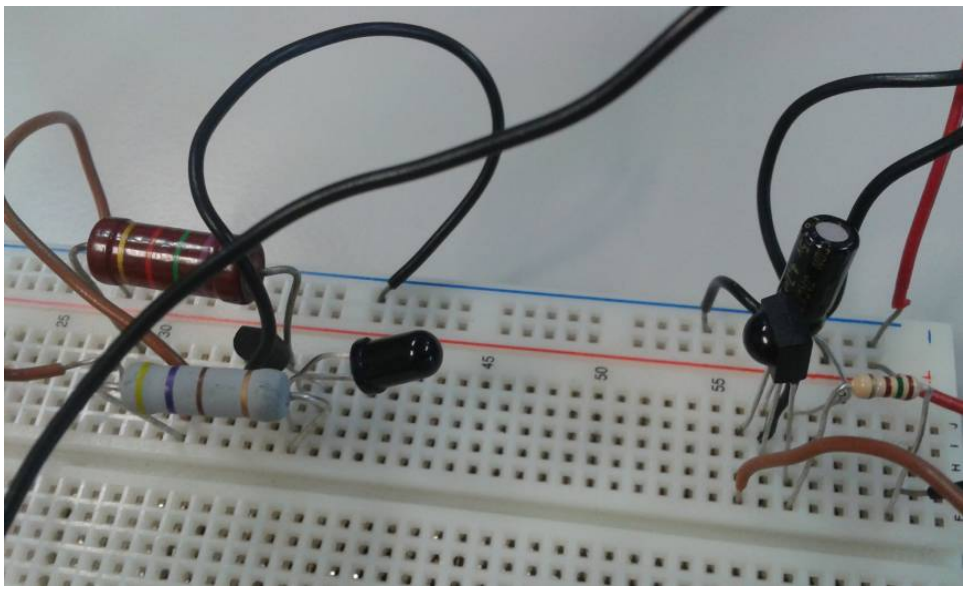


Figura 41. Circuitos acondicionadores del emisor y receptor IR.

Una vez está todo montado correctamente, se puede comenzar a comprobar los resultados prácticos.

**NOTA:** Es importante ir comprobando cada bloque del circuito a medida que se van diseñando. En esta memoria se describen los resultados finales, y se explicarán los problemas encontrados a medida que se diseñaban todos los bloques.

Una vez montado todo el circuito completo para ambos jugadores, lo primero que hay que hacer es cargar el código diseñado en las FPGA's. Cuando ya están programadas, el sistema se resetea para obtener los valores por defecto de todas las señales. El primer análisis a realizar es comprobar visualmente lo que se obtiene a la salida de la vga. Se visualiza el tablero del juego con los dos mapas y los barcos que forman la flota. En la Figura 42 se puede observar el resultado real a la salida de la vga.


Figura 42. Tablero del juego visto a la salida de la vga.

El siguiente paso a comprobar es la puesta en funcionamiento del juego. Para ello hay que pulsar el interruptor que lo inicia. El resultado visual se puede apreciar porque aparece la flecha que selecciona los barcos a colocar. En la Figura 43 se puede observar el resultado.


Figura 43. Tablero del juego al comenzar la partida.

El juego está en marcha y ahora es necesario comprobar la primera fase del mismo. En esta fase el jugador puede desplazarse, mediante la flecha, por todos los tipos de barcos que existen. Esto sirve para elegir el barco a colocar y una vez seleccionado, la flecha desaparece y en su lugar aparece un puntero en el tablero correspondiente al propio jugador. En la Figura 44 se puede observar como el contador ha disminuido una vez se ha elegido el barco, y como aparece el puntero.

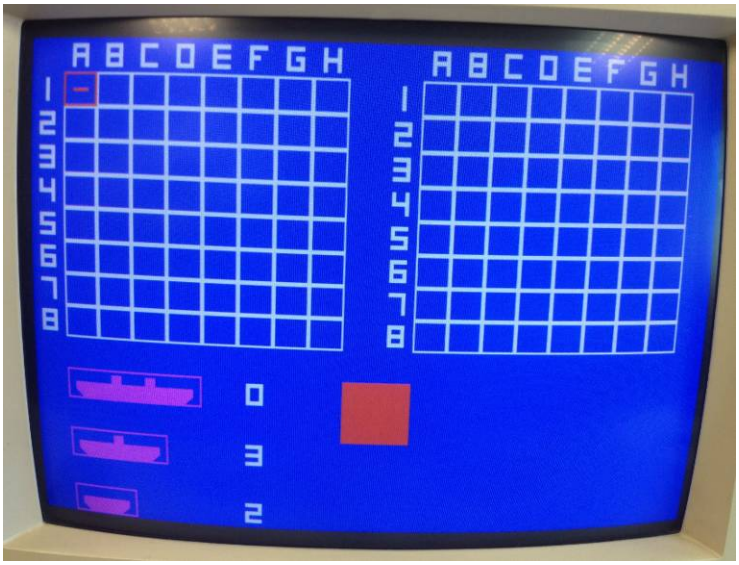



Figura 44. Puntero para colocar el barco en el tablero.

La siguiente validación se traduce en mover el puntero por todo el tablero y comprobar que no es posible salirse de los límites marcados por el mapa. El sistema ignora los movimientos cuando quieren ir más allá de los límites. Se mantiene en la posición cuando: está en la coordenada A y quiere desplazarse a la izquierda, está en la coordenada H y quiere desplazarse a la derecha, está en la coordenada 1 y quiere desplazarse hacia arriba y por último, que está en la coordenada 8 y quiere desplazarse hacia abajo.

Otra comprobación antes de colocar el barco seleccionado en el tablero, es intentar colocarlo en una determinada posición que al situarlo, el barco se salga de los límites del mapa. En este caso, se va a colocar un barco de 4 casillas de longitud, por lo que si se quiere depositar en posición horizontal y a partir de la coordenada F, el sistema no sitúa el barco. Al pulsar *enter,* no se coloca el barco y el puntero se mantiene en la misma posición. Otra opción que se puede utilizar, es rotar el barco a la posición vertical. La forma del puntero cambia y la línea interna del puntero se pone en vertical. En este caso, se habilita el poder colocarlo estando en la coordenada H, pero siempre y cuando no se supere la coordenada 6.

Cuando la casilla se ha elegido correctamente, el barco se coloca en el tablero. En la memoria RAM se introducen todos los datos correspondientes al barco situado, y aparece pintado en el tablero. El sistema vuelve a la fase de elegir otro barco a colocar, por lo que aparece la flecha para volver a elegir. En la Figura 45 se puede observar un barco colocado en el mapa.

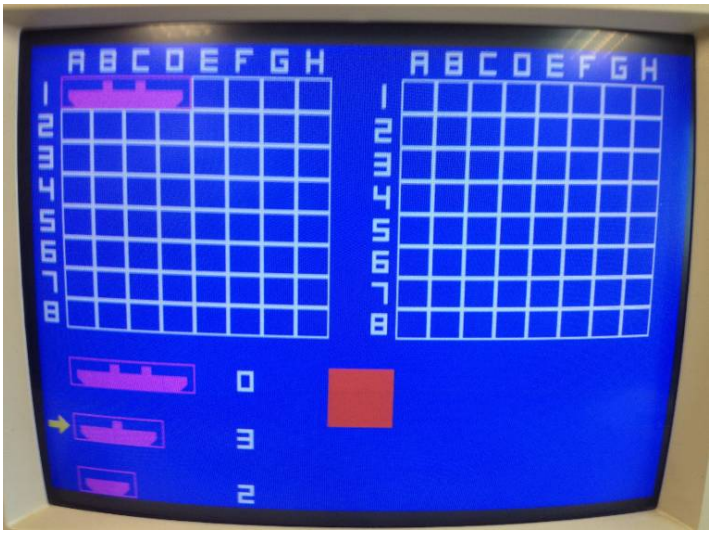



Figura 45. Barco colocado en el tablero.

El procedimiento a seguir a partir de ahora hasta colocar toda la flota es el mismo. Los contadores del resto de tipos de barcos disminuyen a medida que se van seleccionando. Cuando el contador de un tipo de barco llega a '0', ya no se puede elegir ese tipo de barco. La flecha no concede la opción de seleccionarlo y tan sólo se mueve por el resto de barcos que aún se puedan elegir. En el caso expuesto en la Figura 45, no se puede subir y únicamente se pueden seleccionar los marcos medianos y pequeños.

Otra comprobación importante, es intentar colocar un barco sobre otro ya depositado. El sistema analiza todas las posiciones cuando va a colocarlo y si detecta en la memoria RAM que ya hay uno introducido, en cualquiera de las posiciones que vaya a ocupar, entonces no lo deposita.

El primer jugador en finalizar de colocar su flota, recibe el turno inicial y por lo tanto, el cuadrado que asigna el turno se habilita con color verde. En la Figura 46 se muestran todos los barcos colocados en el tablero.

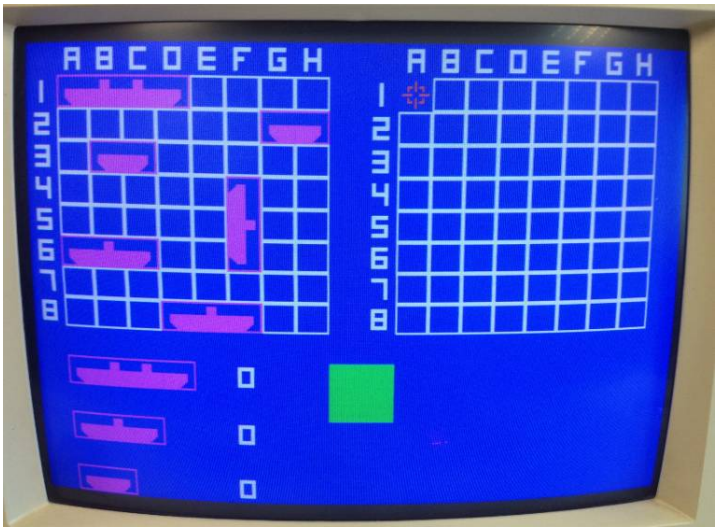



Figura 46. Toda la flota colocada.

Una vez se han colocado todos los barcos, hay que esperar a que el otro jugador termine de colocar los suyos. Este jugador no dispone del primer turno, por lo que su cuadrado sigue deshabilitado en color rojo.

Como ya se ha visto en la imagen anterior, aparece un nuevo puntero en el mapa correspondiente al lanzamiento de bombas (que simula el mapa del jugador contrario). Se ha guardado hasta el último detalle, y al puntero se le ha dado forma de mirilla.

Como ocurría con el tablero de la izquierda, es posible desplazarse por todo el mapa pero sin superar los límites establecidos. Una posible comprobación del correcto funcionamiento del juego, es intentar lanzar una bomba en la casilla que ya se ha disparado otra anteriormente. El sistema lee la memoria RAM1 y comprueba si en esa casilla ya se ha lanzado una bomba, evitando así volver a lanzarla. El puntero se mantiene en esa posición pero no te deja depositarla.

Cuando se ejecuta un lanzamiento, se envía por infrarrojos las coordenadas de la bomba enviada. La comunicación por infrarrojos ha sido una de las tareas más complejas y con mayores problemas del diseño. A lo largo del proceso de depuración cabe destacar uno de los problemas que hubo que resolver por su relevancia.

El problema más importante que surgió fue que no se sincronizaban correctamente las ondas. El sistema se perdía continuamente a la hora del intercambio de bombas. Consideraba que había recibido una señal o que no lo había hecho cuando en realidad era lo contrario. En un principio, se estudio el envío de

datos por parte del emisor del sistema, porque las primeras prácticas se realizaron con un mando universal que simulaba el emisor, y después con el trasmisor final.

Una vez se descartó que los problemas vinieran por parte del emisor, se estudió que la máquina de estados creada para el receptor no recogiese bien los datos que se le enviaban. Un paso llevo a otro, y se demostró que en el bloque del árbitro era necesario un estado que recogiese un pulso de reloj entre las transmisiones de las coordenadas X e Y para sincronizarlas. Pero el problema continuaba y el sistema seguía perdiéndose. Resultó muy complicado saber dónde estaba el error, puesto que el resultado visual mostrado por pantalla no da evidencias de donde puede provenir el fallo.

Finalmente, con la ayuda del osciloscopio y de mucho trabajo, se averiguó que la onda que el receptor recibía no tenía el periodo que se había estudiado teóricamente. Sino que su periodo era mayor (un periodo de 1,90ms frente al teórico 1,78ms), desestabilizando así el diseño creado para interpretar los datos que le llegaban al receptor. La solución al problema fue aumentar la cuenta de un contador (contador de 80000 que afecta a la FSM del primer bloque del receptor), en el estado *Espera_flanco,* para que no se saltase los flancos que se tenían que detectar. El sistema saltaba al estado de Reposo si en 15000 ciclos de reloj no se detectaba ningún flanco, y en realidad necesitaba 20000 ciclos puesto que el periodo de la onda es mayor, y los flancos llegaban más tarde.

Con este problema solucionado, la comunicación se sincroniza y el juego funciona correctamente. Los jugadores pueden disputar una partida y alternar continuamente lanzamientos de proyectiles sin que el sistema se pierda. A continuación se muestra en la Figura 47 el estado de una partida tras varios turnos.

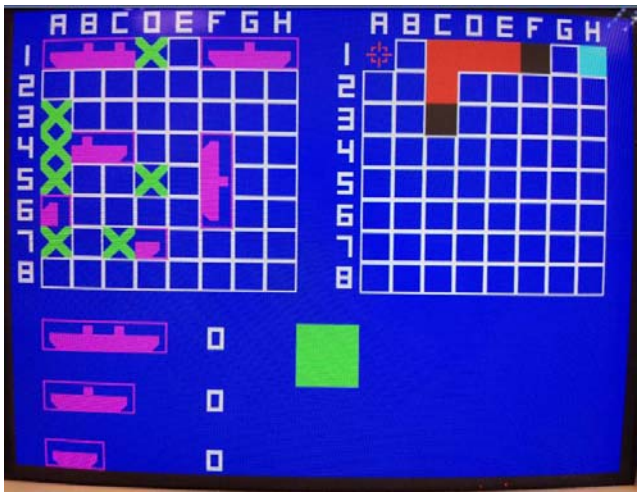



Figura 47. Intercambio de bombas.

Cada vez que un jugador lanza una bomba, el oponente le contesta automáticamente con el resultado del lanzamiento. Si el cuadro es azul significa que no ha impactado en ningún barco (agua). Si por el contario, el cuadro es rojo significa que el proyectil ha impactado en un barco (tocado) y por último, si el cuadro es negro quiere decir que se han dado a todas las posiciones del barco (hundido).

Mientras en el tablero del propio jugador, se van marcando con una X las bombas que el contrario envía.

En la Figura 48 se observa una partida tras varios turnos, mostrando los monitores de ambos jugadores para poder ver con más claridad el buen funcionamiento del diseño. La forma más rápida de interpretarlo es comparar el tablero del jugador Nº1 (monitor de la izquierda), que es el mapa de la izquierda del jugador Nº1, con el mapa de la derecha del jugador Nº2 (monitor de la derecha) y comprobar que la comunicación existe. El mismo procedimiento se puede seguir si se compara el tablero del jugador Nº2 (los otros mapas restantes). En ellos se aprecia que cada bomba enviada tiene su respuesta sin equivocación y que gracias a esta comunicación, el juego fluye perfectamente permitiendo jugar.


Figura 48. Partida del juego tras varios turnos.

Ya se ha validado el funcionamiento del juego. Ahora solo queda comprobar qué ocurre cuando finaliza la partida. Este paso es el último, y se produce cuando uno de los dos jugadores hunde todos los barcos del jugador contrario. Cuando esto ocurre, se envía una señal infrarroja por parte del perdedor indicando que se ha eliminado

toda su flota y que la partida se ha terminado. En la Figura 49 se puede comprobar visualmente que el sistema responde correctamente al final del juego.

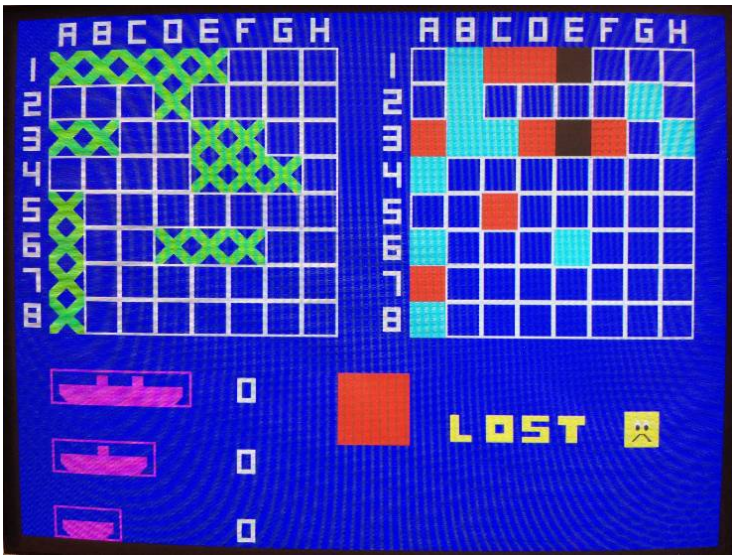



Figura 49. Fin de partida y mensaje para el jugador derrotado.

En el lado opuesto, se encuentra el jugador que ha vencido en la partida y que al recibir la señal infrarroja también muestra un mensaje por pantalla.

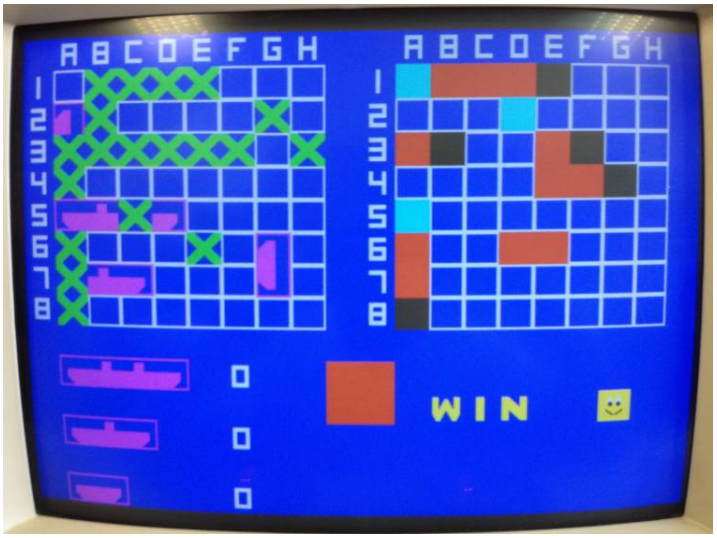



Figura 50 . Fin de partida y mensaje para el jugador vencedor.

En conclusión, se puede asegurar que tras las pruebas realizadas el diseño funciona correctamente. En la Figura 51 se confirma este hecho, y se muestra el sistema completo montado y con una partida terminada.

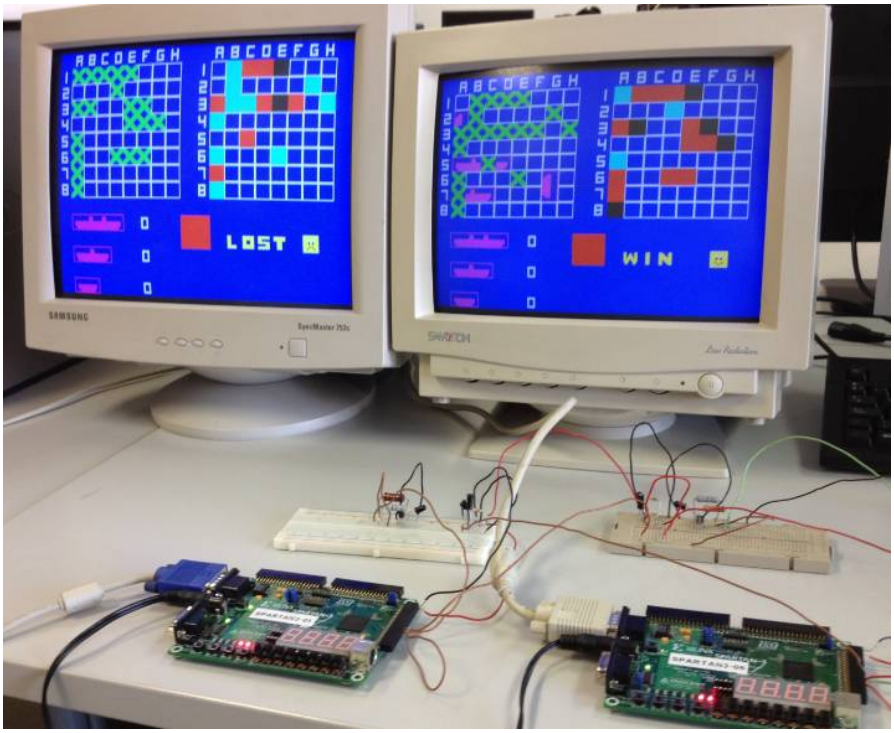



Figura 51. Montaje del sistema con el juego finalizado.

## *4.1 RESULTADOS DE SÍNTESIS DEL CIRCUITO*

Los resultados de la síntesis con ISE para el dispositivo "xc3s500e-4ft256" de la familia Spartan-3 de nuestro diseño son los siguientes:

4.1. ÁREA

- Número de biestables: 943 (24%)
- Número de LUTS: 2120 (55%)

4.2 TIEMPO

- Tiempo mínimo posible empleado por el reloj del sistema: 19,140 ns.
- Frecuencia máxima empleada por el reloj del sistema: 52,246 MHz.

Como se ha utilizado una FPGA de bajos recursos y aún así no se han utilizado por completo como se puede observar con los resultados obtenidos, se deduce que no habrá ningún problema si en un futuro se mejora el sistema. Por otro lado, la frecuencia del sistema está por encima de la frecuencia de la placa FPGA (50MHz), por lo que es otra razón que confirma que el sistema funciona correctamente.

# CAPÍTULO 5

## 5 CONCLUSIONES Y TRABAJOS FUTUROS

El objetivo principal de este proyecto era diseñar en VHDL un sistema digital que simulase el juego de Hundir la Flota, y que estuviese implementando en una FPGA. Tras realizar el diseño y observando los resultados obtenidos, se puede afirmar que se ha alcanzado el objetivo satisfactoriamente.

Con el diseño del emisor y el receptor de infrarrojos se ha conseguido establecer una comunicación entre dos usuarios. Por ello, se ha logrado otro de los objetivos de este proyecto, que el juego sea interactivo. Además el sistema es capaz de contestar automáticamente al oponente cuando se le han enviado las coordenadas de una bomba. Por razones como ésta, se ha alcanzado una comunicación muy fluida convirtiendo el juego en un entretenimiento asegurado.

El diseño cumple las reglas del juego original y se puede disputar una partida hasta el final sin tener problemas de que el sistema se quede colgado. Es importante comentar esto, puesto que se ha comprobado empíricamente que es real. En el supuesto caso en que hubiese un fallo en la comunicación y el sistema se perdiese, obligando a terminar la partida, se debe comprobar el correcto funcionamiento de los dispositivos conectados en los circuitos porque es muy probable que sean los causantes del fallo.

Después de realizar el diseño se ha estudiado si es factible incluir este juego en las prácticas de la asignatura "Laboratorio de Microelectrónica". Para realizar el estudio se ha tenido en cuenta tres cosas: Nº de personas que forman el grupo de trabajo, tiempo para llevarla a cabo y conocimientos previos del alumno en la asignatura. A su vez, se propone dividir la realización del sistema en una parte básica obligatoria (para el aprobado) y otra opcional (para subir nota):

- Trabajo básico:
  - Representar el tablero (implementar bloque VGA).
  - Elegir un cierto número de barcos de un solo tipo.
  - Diseñar el control del emisor/receptor de IR para enviar las señales: Fin de colocación de todos los barcos que determina el turno inicial de la partida, posición de una bomba, respuesta al lanzamiento de una bomba y fin de la partida.

- Diseñar funciones de control: elegir y colocar barco, jugar y fin.
- Implementar la memoria del juego.
- Implementar la interfaz del juego.

- Trabajo opcional (mejoras):
  - Mejorar la interfaz gráfica del tablero: barcos, coordenadas en tableros, mensaje final de partida, etc.
  - Considerar barcos de distintos tipos y diseñar la función para moverse y seleccionar un barco entre los distintos tipos.

Como trabajo futuro se propone diseñar y fabricar una PCB para los circuitos emisor y receptor de infrarrojos. Con esto se pretende mejorar el sistema de comunicación y evitar posibles fallos en el montaje. El resultado final debe integrar ambos circuitos y dar acceso únicamente del conexionado al resto del sistema.

# CAPÍTULO 6

## 6 PRESUPUESTO

A continuación se detallan los costes globales de la realización de este Proyecto Fin de Carrera. El presupuesto global se divide según el tipo de coste. En la Tabla 2 se muestra el presupuesto en material necesario.

### ❖ Coste de material

| Concepto | Coste unitario | Cantidad | Total |
|---|---|---|---|
| Ordenador de sobremesa | 75€ | 1 | 75€ |
| FPGA Spartan-3 Board | 130€ | 2 | 260€ |
| Monitor con vga | 100€ | 2 | 200€ |
| Placa protoboard | 9,20€ | 2 | 18,40€ |
| Receptor de infrarrojos SFH 5110-36 | 0,73€ | 5 | 3,65€ |
| Emisor de infrarrojos SFH 415-U | 0,29€ | 5 | 1,45€ |
| Componentes electrónicos | 4€ | 2 | 8€ |
| | | **Total:** | **566,5€** |

Tabla 2. Coste del equipo.

El valor del ordenador de sobremesa está amortizado, ya que su tiempo de vida son 5 años (60 meses) y el tiempo de utilización en el proyecto han sido 5 meses. Su coste total es de 900€ pero el valor amortizado para esos meses es de 75€.

En esta sección se incluye el coste de personal involucrado en su desarrollo durante el tiempo del mismo. Este periodo ha sido de 10 meses durante unas 4 horas de jornada, lo que es equivalente a 5 meses con una jornada de 8h. En la Tabla 3 se puede observar el presupuesto empleado para el personal. Mientras que en la Tabla 4 se puede observar el presupuesto empleado en el software.

## ❖ Coste de personal

| Concepto | Coste unitario | Cantidad | Total |
|---|---|---|---|
| Ingeniero Técnico Industrial | 2694,39€/mes | 5 meses | 13.471,95€ |
| | | **Total:** | **13.471,95€** |

Tabla 3. Coste de personal.

## ❖ Coste de software

| Concepto | Coste unitario | Cantidad | Total |
|---|---|---|---|
| ISE de Xilinx | Licencia gratuita | 1 | 0€ |
| ModelSim | Licencia gratuita | 1 | 0€ |
| Microsoft Office 2007 | 200€ | 1 | 200€ |
| | | **Total:** | **200€** |

Tabla 4. Coste de software.

## ❖ Coste total del proyecto

El importe total del desarrollo asciende a:

| Concepto | Total |
|---|---|
| Coste de material | 566,5€ |
| Coste del personal | 13.471,95€ |
| Coste del software | 200€ |
| Coste indirecto | 2847,69 |
| **Total:** | **17.086,14€** |

Tabla 5. Coste total del proyecto.

# CAPÍTULO 7

# 7 BIBLIOGRAFÍA

- "Diseño de circuitos digitales con VHDL" Luis Entrena; Mario García; Celia López y Marta Portela, UC3M 2005
- "HDL Chip Design" Smith Douglas, 1998

## *7.1 REFERENCIAS*

### MANUALES

[1] Mario García y Fernando Casado "Manual de Laboratorio de Microelectrónica", Departamento de Tecnología Electrónica UC3M, 8 de Octubre de 2007

[2] Luis Mengibar y Fernando Casado "Manual Xilinx ISE", Departamento de Tecnología Electrónica UC3M

[3] Luis Mengibar y Fernando Casado "Manual básico de uso de la herramienta ModelSim", Departamento de Tecnología Electrónica UC3M, 9 de Octubre de 2006

### DOCUMENTOS

[4] Xilinx "Spartan-3E Starter Kit Board User Guide", 4 de Diciembre de 2009

[5] Paul Seerden "Application Note of an 87LPC76x microcontroller from Philips Semiconductors", 2003 May 15, (http://www.nxp.com/documents/application_note/AN10210.pdf )

[6] Fernando Remiro Domínguez "Mando a distancia por infrarrojos", I.E.S Juan de la Cierva

[7] OSRAM Opto Semiconductors GmbH "SFH-415 Datasheet", (http://catalog.osram-os.com/catalogue/catalogue.do?favOid=0000000200005efb001c0023&act=showBookmark ) 21 de agosto de 2009

[8] "BC547 Datasheet" http://www.datasheetcatalog.com/datasheets_pdf/B/C/5/4/BC547.shtml )

[9] OSRAM Opto Semiconductors GmbH "SFH-5110 Datasheet" (http://catalog.osram-os.com/catalogue/catalogue.do?favOid=00000002000246c8001c0023&act=showBookmark) 6 de junio de 2008

## DIRECCIONES DE INTERNET

[10]http://www.ucontrol.com.ar/Articulos/protocolo_de_los_controles_remotos_philips_RC5/protocolo_de_los_controles_remotos_philips_RC5.htm 10 de abril de 2007